Recommendations in Visual Analytics Using Emotions

A Mixed-Initiative Interaction Approach

by

Prateek Panwar

A thesis submitted to the School of Graduate and Postdoctoral Studies in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science

Faculty of Science
UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY
Oshawa, Ontario, Canada

JULY 2018

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This thesis is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements.

Prateek Panwar
July 2018

Acknowledgements

I would like to express my gratitude to my supervisor, Dr. Christopher Collins, for his continued guidance and support over the course of my graduate studies. Dr. Collins believed in me and always encouraged me to explore innovative ideas for solving real-world problems.

Also, I would like to acknowledge all the members of Vialab for their encouragement in the gloomiest days and valuable comments during the development of this thesis.

Finally, I must express my very profound gratitude to my mother and my sister for always being supportive. This accomplishment would not have been possible without them. Thank you!

Abstract

The thesis demonstrates an idea for helping users in visual analytic tasks by investigating some critical steps required for providing recommendations. The proposed model uses mixed-initiative interaction approach by detecting users' negative emotions, caused by the visual analytic tasks, as a cue to generate useful guidance. For building a negative emotion detection classifier, I have created a dataset from 28 participants carrying out intentionally difficult visualization tasks and collected their emotional responses using multiple biosensors. I used this dataset to built a real-time emotion detection model which predicts mental state in every 4s. Next, the visualization tool uses the detected emotions to generate a recommendation and decide when to intervene. Additionally, the system also adapts intrusion level by analyzing long-term emotions, and decide the best way to show the help. Finally, I have concluded this work by discussing the design space of interventions for providing just-in-time assistance in visual analytics.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Data visualization has been around for decades and is gaining a lot of attention in every area, such as medical, business and text analysis, throughout the world [12]. Visualizing a big dataset helps users to understand patterns, allows them to interact with data and calculate the key findings visually. In addition, analyzing data from the past can help in making decisions based on the trends or in predicting the future outcomes. Figures 1.1 and 1.2 show an interactive text visualization interface for analyzing patterns in a document and a health activity visualization respectively.

The complexity of data visualization is dependent on the tasks and can vary from a simple bar graph to a more complicated design for solving real-world problems like predicting a natural disaster based on the historical data. Again, as can be seen in Figure 1.1, a clear representation of a large text document helps users understanding something which was not visible in the original text file and allows them to see the trends in the text. Information visualization is used by both non-expert users (people who don't work in a data science field) and expert users. For example, non-experts deciding between which car to buy based on multiple attributes such as budget, model, type etc. (Figure 1.3); and experts analyzing big data for companies, predicting stock

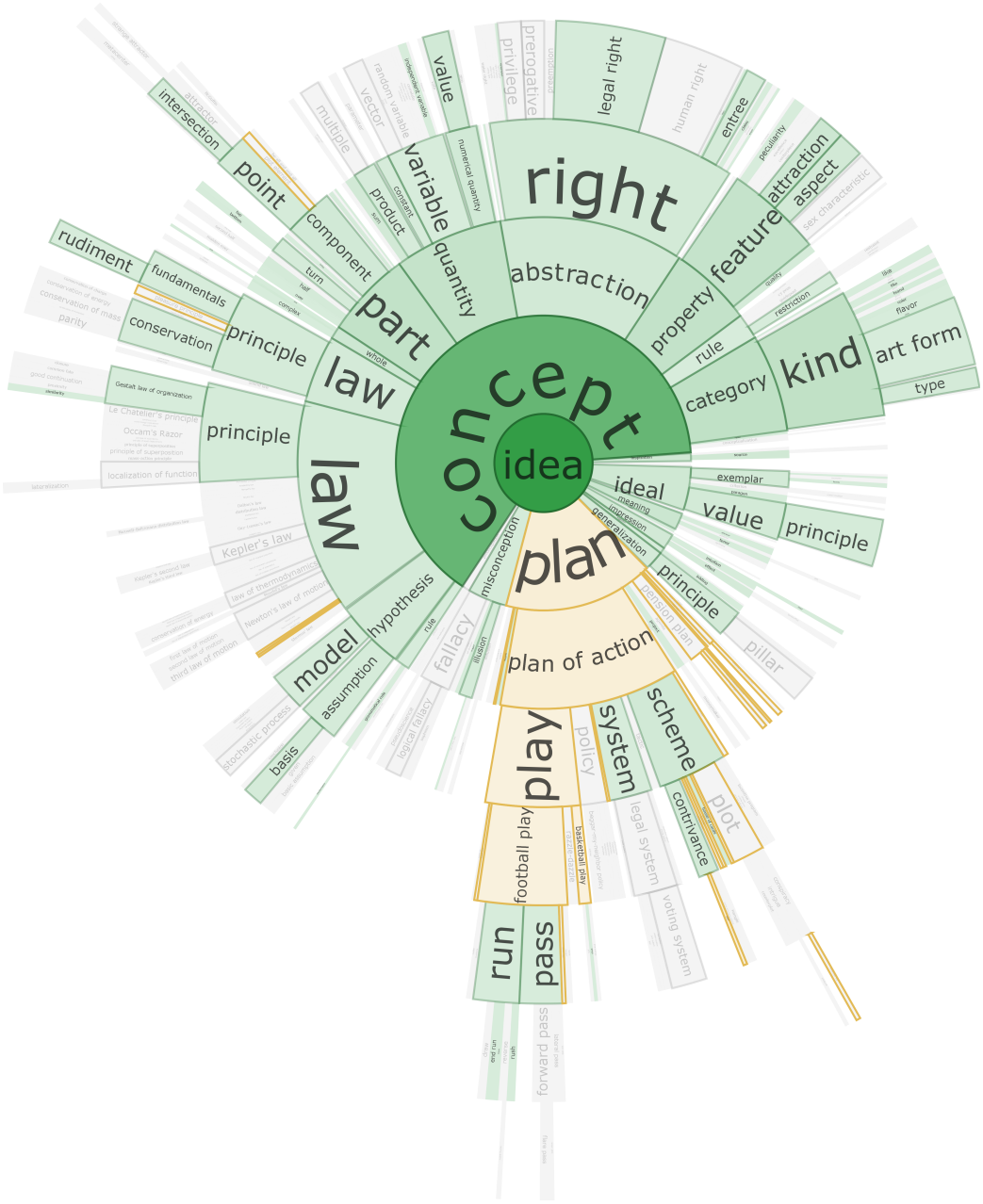

Figure 1.1: DocuBurst interface for analyzing contents of long documents [6].

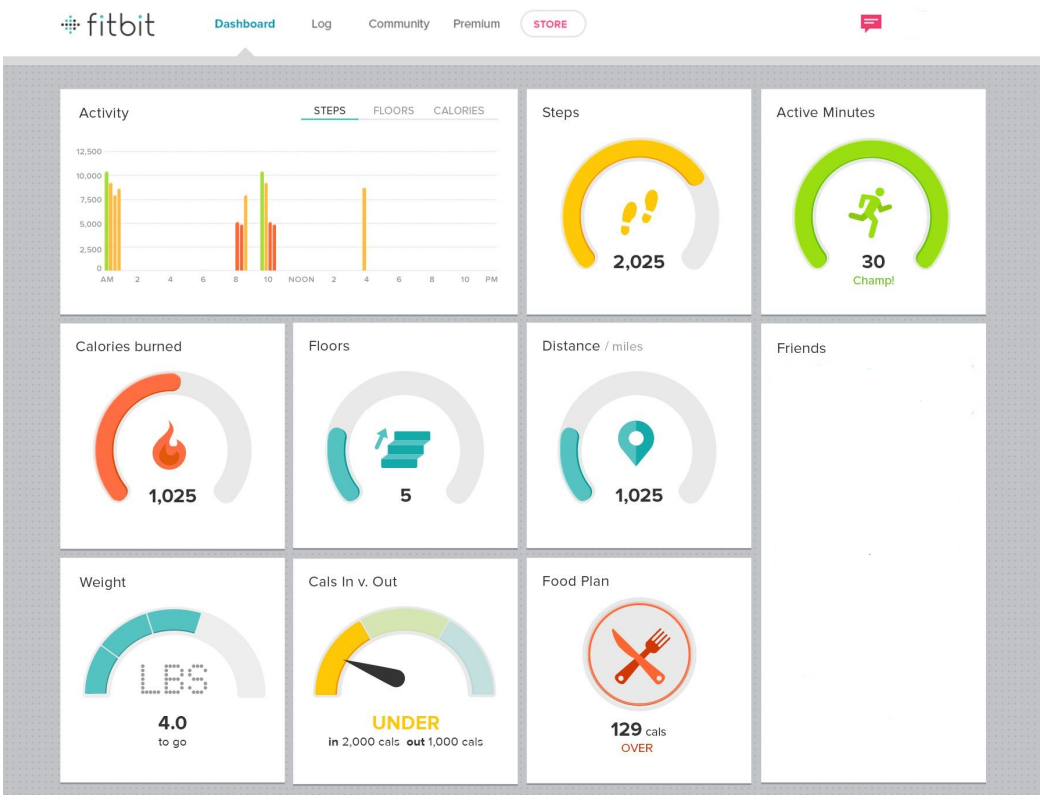

Figure 1.2: Fitbit activity visualization (fitbit.com).

pricing, understanding text or in academics.

Although information visualization is very helpful, it is also a challenging task [20, 44]. Some of the significant challenges are: (1) dealing with a complex dataset involving multiple variables; (2) using a new or unfamiliar dataset where the nature of the data is unknown; (3) using a new tool or merely a new visualization. Any of these cases requires an extra effort from the user which could result in a high cognitive load. Moreover, working in the real world scenario where deadline pressure, work stress and personal life crises are already affecting the performance of a user [10], it becomes difficult to stay focused while solving these data analysis tasks.

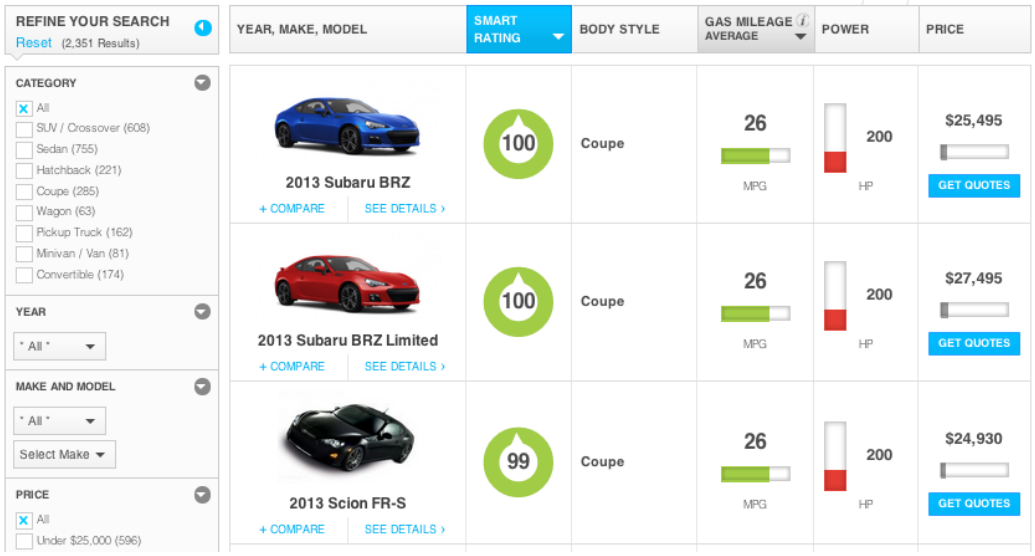

Figure 1.3: Comparing a car based on multiple attributes [37].

All users, expert and non-expert, go through a range of negative emotions such as confusion, frustration and anger; when they get stuck in a task, and this increases the chance of making more errors or disengagement [4]. For example, if a user is new to a particular visualization tool and got stuck while doing a task due to the complex user-interface (UI), then the user will go through a range of negative emotions because this complexity is blocking the workflow and distracting the user's attention. Therefore, after a certain amount of time the user will feel demotivated and ultimately disengage [4]. In other words, these negative emotions might be an indication of a hindrance in the workflow which could lead to mind wandering or disengagement [2]. Moreover, there are many existing visualization tools available which help in facilitating the data analysis process such as Tableau, Microsoft Power BI and Google Analytics (Figure 1.4), but these interfaces also display many functions which can be overwhelming and confusing for the users.

There is a need to build a support system which would provide users with meaningful recommendation or guidance and help them to overcome issues

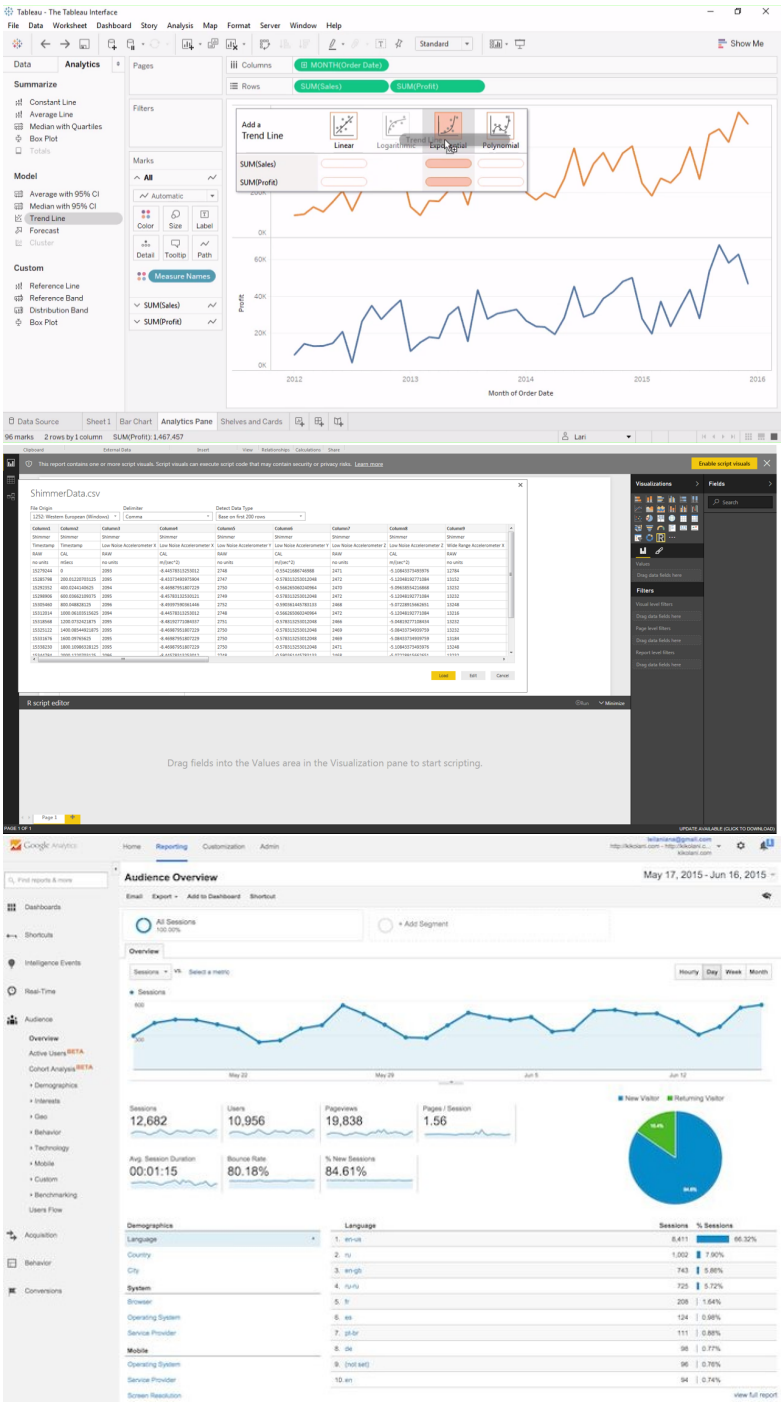

Figure 1.4: Existing data visualization tools have many interactive capabilities which may be confusing to users. Top: Tableau; Middle: Microsoft Power BI; Bottom: Google Analytics.

in understanding the data and hence, prevent disengagement. Providing meaningful recommendations in a visual data analytic tools have been a challenging task because: (1) The recommendation system doesn't know when is the right time to provide help and continuously showing help on the screen would cause distraction and could be annoying; (2) If a user is stuck and feeling confused, the system doesn't know the cause of this emotion and how to fix it as same emotion can occur for different cases (interface-related, data-related or external factors); (3) System cannot predict intensity of an emotion and therefore, failed to vary the intrusiveness accurately. Moreover, the overall interaction is still one-sided, and the system doesn't understand the mental state of users or how to react differently to different states. This work argues that to build a smart recommendation system; there is a need for leveraging mixed-initiative interaction approach so that the system could understand the cues from the user and react accordingly (bi-directional interaction).

A recommendation tool called Clippy was introduced in Microsoft Office, but in a short period, it was discontinued due to poor feedback from the users. After the discontinuation, many studies and articles [15, 23, 45] found that the problem with Clippy was its intrusiveness and providing help which neither meaningful nor related to the task. The example demonstrates the need to improve two-way interaction, also known as mixed-initiative interaction, for building a better recommendation system which could understand what a user needs and guide them accordingly while also taking intrusion level into account.

1.1 Motivation

Recommendation systems have been extensively studied and are successfully implemented in many different areas; for example, in games [31], e-commerce [36], and tutoring systems [48]. Some common examples of recommendations sys-

tems which we use in our daily life are video suggestion on YouTube, movie recommendation on Netflix, friend suggestion on Facebook, and Amazon "what you might like" product suggestions; these systems are getting smarter every day with the boom in machine learning technology [41]. Usually, these systems construct a user profile for each individual and use implicit or explicit feedback from the users to learn about their preferences. Here, implicit feedback is information that gets collected in the background for deducing user's behaviour such as monitoring user's mouse movements, history or time spent on a specific page; whereas, explicit feedback collects the data by prompting users to provide particular information like rating a product or filling a survey.

As users can't infer implicit feedback because they might not necessarily be aware of it, this type of feedback has proven successful in tutoring systems and websites for forming a useful recommendation in comparison to the explicit feedback which can be subjective and biased [18].

1.1.1 Adaptive Systems

There are many existing tutoring software and video games which adapt to user's learning curve [4, 13, 42]. Moreover, they use different prediction models, ranging from simple classification models like k-nearest neighbours to more complex classifiers like neural networks. These models run in the backend and use implicit or explicit feedback from the user for deciding an appropriate action in real-time. Although these techniques have been applied to visual analytic tasks [25, 42] but never have tried to use emotions as a feedback for the recommendation systems, to best of my knowledge.

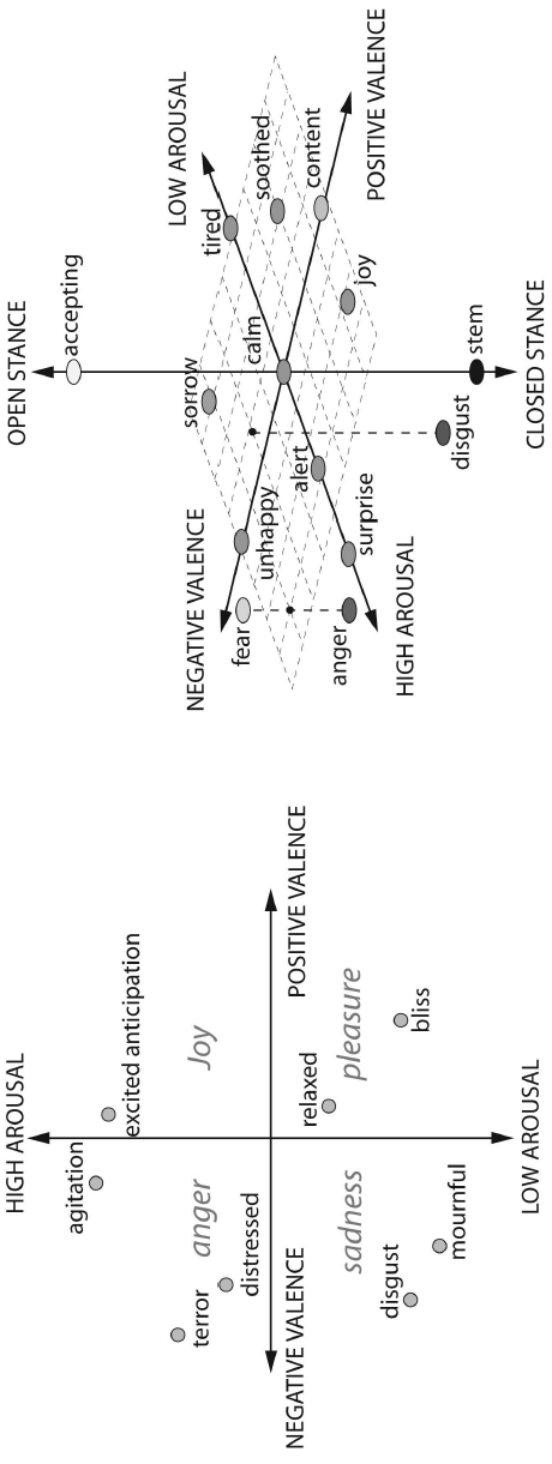

Figure 1.5: Visualizing emotions on valence and arousal scale. Left: Two-dimensional model by valence and arousal. Right: Three-dimensional model by valence, arousal and stance [21].

1.1.2 Affective Computing

Emotion detection and user behaviour analysis got its roots from psychology, but R. Picard first introduced the affective computing term in computer science in 1995 [33], and since then researchers are applying this technique in different areas of computer science [3]. Affective computing is the study of detecting or interpreting human emotions by reading patterns or signals emitted by the body, for example, blinking rate and gestures. These signals or patterns are often recorded with the help of bio-metric sensors such as an eye tracker or electroencephalography (EEG) [26, 51]. The affective computing field comes under the family of cognitive science which is the study of mind and its processes.

The fundamental idea to recognize an emotion is by calculating the values of *valence* and *arousal* using bio-senors. Arousal is the intensity of emotion and valence helps to distinguish between positive and negative emotion. Different values of arousal and valence are the measure of different emotions, for example, positive values of valence and arousal represent positive emotions such as joy and excitement. Figure 1.5 visualizes a 2 dimensional and 3 dimensional emotion graph which explains how a basic emotion (in 2D graph) or more complicated emotions (in 3D graph) varies with valence and arousal values [21].

Furthermore, bio-sensors record two types of signals — physical and physiological, and allow researchers to observe the patterns and distinguish different mental states. The human body reacts differently with every emotion such as in case of anger, heart rate increases with an increase in blood pressure and breathing rate. These signals help in calculating valance and arousal levels.

Physical Signal can be captured from body movements for example — body gestures, muscle movements, and gaze movement. Recent studies have shown that these signals can be beneficial to detect user state. People tend to show some common patterns for particular mental states. For example, a study by

McCuaig et al. [28] revealed that people tend to tilt their head to the left while frustrated. Moreover, the predicted emotions from the physical signals are easy to verify by video recording the experiment and doing a retrospective think-aloud study or just a simple note-taking and observing the user. Taking the same example by McCuaig et al. [28], tilting head was an indication of frustration which was later verified by looking at the video. The only drawback with these signals are, physical signals are conscious and can be easily manipulated or suppressed by users.

Physiological Signal are the signals discharged by a human body involuntarily and any changes in these signals mean some reaction or change in mental state. Many researchers have used physiological signals in their studies for detecting emotions as these signals are continuous and unconscious; that is, they can't be manipulated by the participants. Electrical signals firing by neurons, skin conductance and heart rate are some of the examples of physiological signals; but filtering the noise from these signals is one of the main challenges. Another challenge is ambiguity; it difficult to prove that the changes recorded are from the test session or another reason such as fluctuation in the room temperature. Therefore, it is essential to keep the environment controlled through-out the session to record accurate data. Moreover, to overcome these limitations, the physiological signals are used in combination with other methods for justifying the predicted emotion. For example, if the signals predicted that a user is feeling angry while performing a task then tracking gaze activity could confirm that this anger was caused by the given task and not by any change in the experiment environment. Some of the most commonly used signal are Electroencephalography (EEG), Galvanic skin response (GSR) and Electrocardiography (ECG).

Using different bio-sensors to track these body signals has shown potential in accurately detecting mental states (details examples are in the next chapter). Additionally, previous research has also tested prediction models for on-the-fly for emotion detection [26]. Though, placement of these devices on a

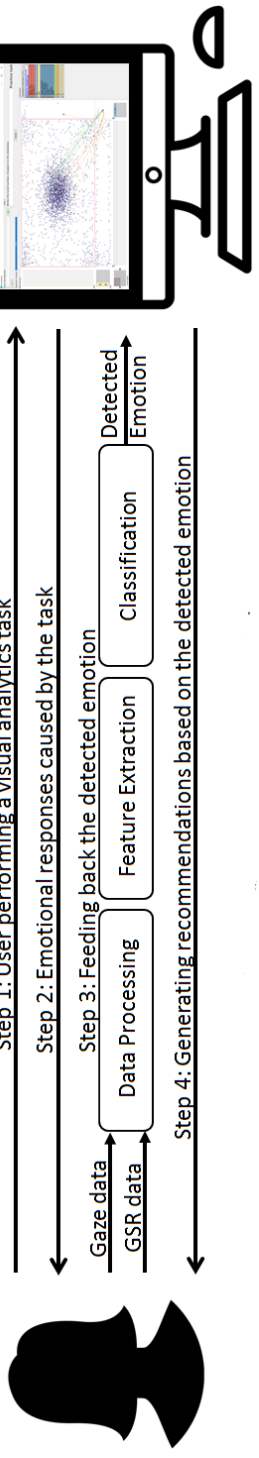

Figure 1.6: Work-flow of the proposed model.

user's body has been shown to be distracting in the tasks [46]. As the technology is advancing, the size of these sensors shrinking rapidly, for example, ECG sensors used in Fitbit device. I believe that blending these techniques in information visualization could help in developing a better recommendation system for visual analytic tools.

1.2 Contribution

After observing the gaps in recommendation system design and work in affective computing, I propose a recommendation model in visual analytics that would detect the negative emotions of the user, analyze it and provide meaningful guidance in real-time to prevent disengagement. Figure 1.6 gives an insight of the work-flow of the proposed system, and the contribution of this thesis leads to:

- Choosing the least intrusive bio-sensors which will minimize the users' distraction caused by the complicated setup or calibration process. Also, these devices should still be able to provide enough information for negative emotion detection.
- Next, developing a real-time negative emotion detection algorithm using the selected sensors. This thesis focuses on detecting frustration and any further mentions of the term *negative emotion(s)* is referred to detecting frustration state.
- Helping users in visual analytic tasks by providing meaningful recommendations in real time using human emotions as implicit feedback.

For investigating the last contribution point, I followed up the idea mentioned by Conati et al. [7] and Olmo et al. [8] in adaptive visualization interface context and applied it on building a smart recommendation system. The idea

served as a foundation for this thesis which explored all the dimensions mentioned below.

1.2.1 When to Show Help

Real-time detection for predicting user's mental state, here negative emotions, and feeding back into the analytic system to find out when a user needs help or when is the time to show help so it won't annoy the user.

1.2.2 What Type of Help Should be Shown

For an effective and useful recommendation, detecting the source of negative emotion is very important. For example, a user can be frustrated by either the interface functions or dataset complexity. Knowing this would help the system to decide what kind of help should be provided — interface related or dataset related. I leveraged gaze location information to differentiate between these cases. Details are discussed in the later chapters.

1.2.3 How to Show the Help

After knowing when and what help a user needs, it is also important how to show it so that it would aid the users in solving the task or understanding the dataset and helping them to overcome the negative emotion. For example, if the system detects that a user is confused because of the interface functionality, the next step is "how to help the user?". Would it be helpful to highlight some interface buttons, or pop up a message box, or open up the interface manual? There are ample possibilities to show the help but finding out which way would be the best for users, so that they won't get annoyed by it, is a challenge. Therefore, I explored this topic and discussed some necessary steps to

consider like the degree of intrusion and design space of recommendations, before providing a recommendation.

1.3 Organization

Chapter 2 discusses related work on the emotion detection techniques, adaptive systems and challenges in recommendation systems. Chapter 3 describes the user study design and data collection process. The details about the data processing and classification model testing are discussed in Chapter 4 followed by designing the real-time negative emotion detection in Chapter 5. Chapter 6 explains fundamentals of creating a recommendation system. Finally, Chapter 7 addresses limitations of this work, discusses the results and concludes the thesis with ideas for future work.

Chapter 2

Related Work

This chapter gives an insight of previous related work done in affecting computing and recommendations in visualization which inspired this thesis.

2.1 Affective Computing

Past research in affective computing has investigated detecting user's mental state such as mind wandering, stress and frustration. Bixler et al. [2] demonstrated a novel technique for detecting mind-wandering (MW) using eye gaze information. MW is one of the outcomes of negative emotion, and this is what I wanted to detect and prevent before users become disengaged. Similar to the motivation of my work, the authors attempted to sustain the engagement period by reducing the factors that could potentially cause MW. Moreover, the paper talks about the idea of creating intelligent systems that could use this information to intervene and restore user's attention to the respective task. For building the MW classifier, gaze data were collected from 178 participants in an on-screen reading tasks (see Figure 2.1). Three sets of gaze features were used for supervised classification — 30 global features, 19 local features, and

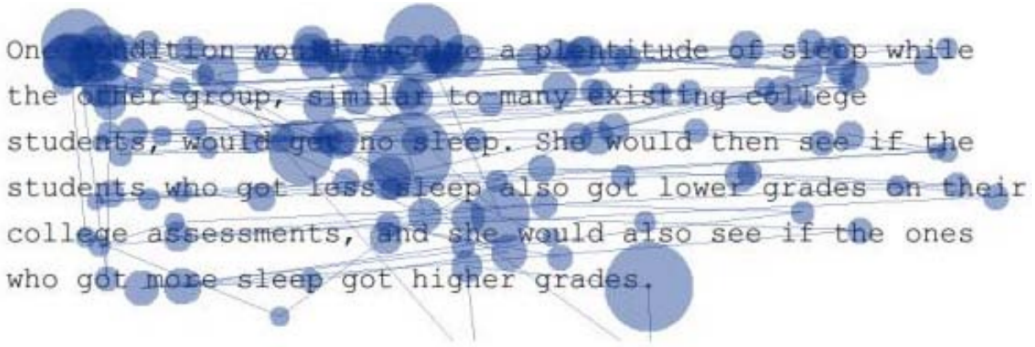

Figure 2.1: Gaze pattern from reading, showing pauses and re-reading [2].

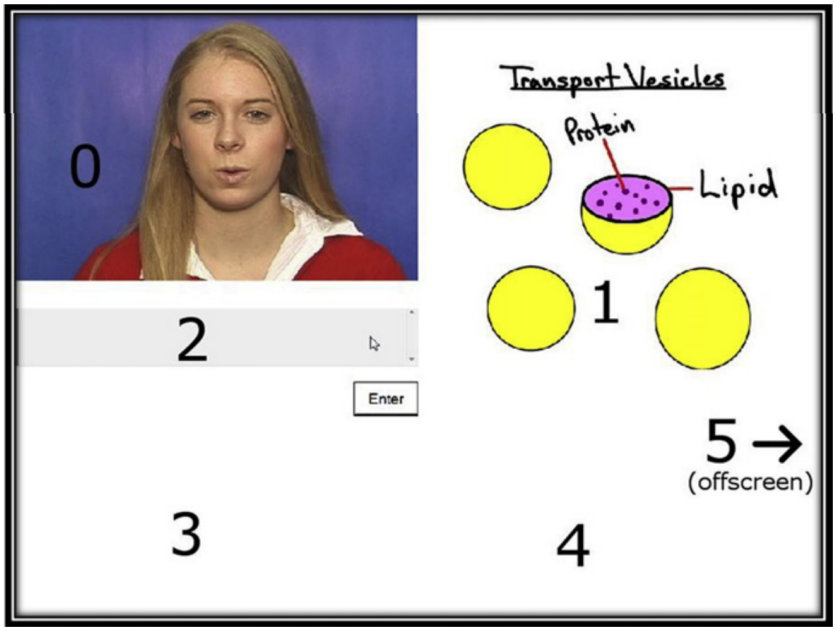

Figure 2.2: Interface used for teaching by D'Mello et al. [9]. The numbers represent different gaze zones in the interface.

12 contextual features yielded at 72% accuracy in detecting MW.

Similarly, gaze information was used in intelligent tutoring systems (ITS) to detect students' boredom and disengagement dynamically. Later, the authors used gaze-sensitive dialogue based interventions to re-orient the user's attention towards the tasks [9]. They did a controlled experiment with 48 students on four biology topics — two on gaze reactive and two on non-gaze versions of the tutor. Figure 2.2 shows the tutoring system that was used in the user study. In the Figure, zone 0 shows the animated tutoring agent, zone 1 shows the pictures related to the topic, zone 3 and 4 were the blank area, zone 2 was a text box, and zone 5 was an off-screen area. The interface can have 15 different responses based on the gaze patterns on different zones. To trigger the gaze-reactive interventions, the authors used a rule-based approach where one of the conditions was that if the user hasn't looked at the zone 0 and 1 for more than 5s. One of the dialogues the agent could say was "Please pay attention." After each lecture, the participants were required to self-report their attention level on a post-lecture engagement questionnaire. The results indicated that gaze-sensitive dialogues were successful in re-orienting the attention.

These two papers gave me some insight into how an eye tracker can be helpful in determining user's attention on the screen. Moreover, I also concluded that recording gaze data is least intrusive as the eye tracker doesn't need to be attached to the user's body and doesn't require any complicated calibration.

Work by Conati looks over human factors in visualization and recommendation systems and has also inspired this thesis. Her work with Steichen [38, 39] have proposed a method which looks over adapting aspects of visualization with every individual by measuring user's cognitive abilities and predicting performance from eye gaze data. The research questions this article tried to answer were: (1) To what extent can a user's current task, performance,

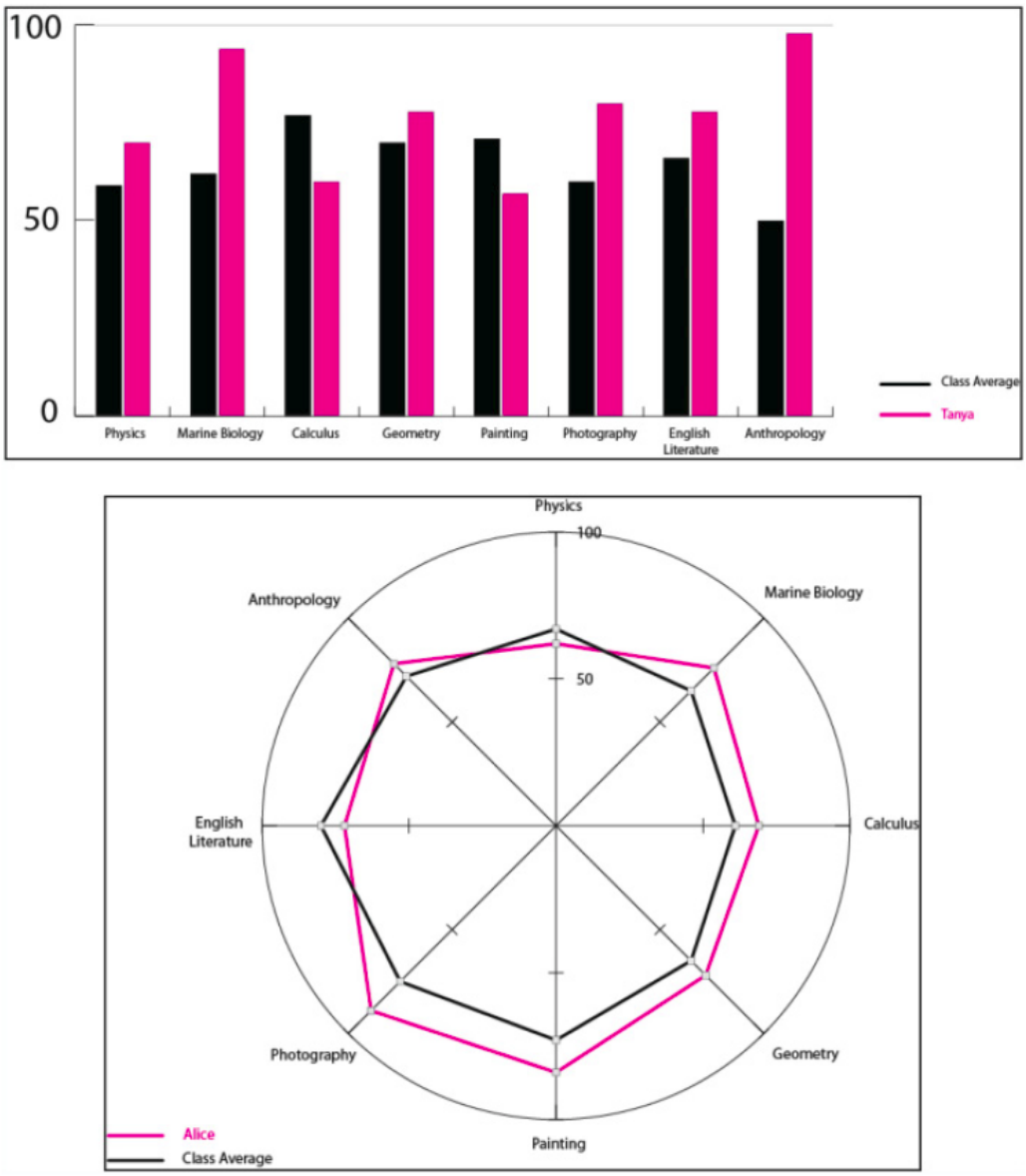

Figure 2.3: Bar graph and radar graph interface used by Steichen et al. [39] for studying gaze path on different areas of the interfaces.

long-term cognitive abilities, and visualization expertise be inferred from eye gaze data? (2) Which gaze features are the most informative?

Moreover, the authors talked about how the visualization interface can change its functions and provide user-specific support. Additionally, they investigated gaze area of interest for finding which part of the interface should be changed to support the user's cognitive abilities. For example, if a user with low cognitive abilities is making a high number of gaze transitions on a particular part of the interface, then the system should be able to provide support related to that area. For this, they conducted an experimental study with 35 participants (18 females) where the participants were required to perform tasks (with varying difficulty and types) on two basic visualizations — bar graphs and radar graphs (Figure 2.3). The tasks were to evaluate the performance of one or two students in eight different academic courses on an artificially generated dataset. Later, 74 features were extracted from the gaze data which included fixation and saccade features such as the number of fixations and saccade length, and area of interest related features like time spent on each interface area and most extended fixation in each area. To test different models on the list of features, the WEKA machine learning toolkit was used, and the accuracy was discussed separately for each task type and complexity (total 5 types and 9 complexity). Finally, the authors found that linear regression model consistently achieved the highest accuracy but overall accuracy for each category and complexity was in the range of 55% to 60%. Also, the results indicated that the contribution of features changes with every task goal but the area of interest related features were most crucial.

Another work by Jaques et al. [19] demonstrates a novel technique of predicting user's affect (boredom and curiosity) in an ITS system using gaze data. Here, the authors ran a user study with 67 participants and collected the gaze data while the participants performed given tasks on an ITS system called MetaTutor. After completing the tasks, the participants self-reported their emotions using the Emotions-Value questionnaire (EVQ) which was based on

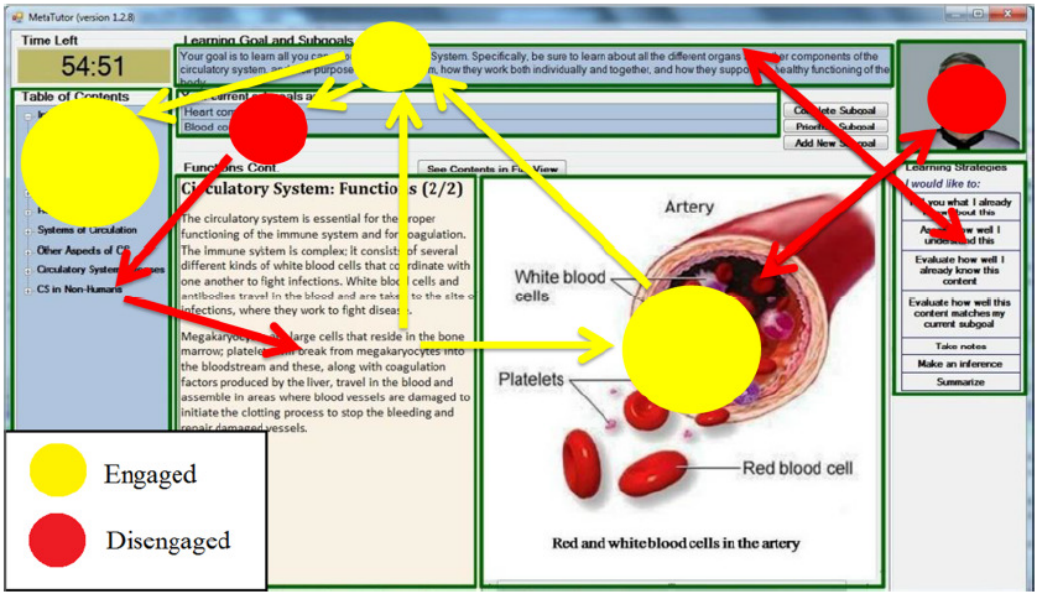

Figure 2.4: Depicted gaze trends for engaged and disengaged students [19].

the Pekrun's Academic Emotions Questionnaire [32]. Next, 166 features were extracted from fixations and saccades, and area of interest. The authors achieved 69% and 73% accuracy for detecting boredom (linear regression model) and disengagement (random forest model) respectively. Figure 2.4 visualizes fixations and saccades in engaged and disengaged students.

A study by Sun et al. [40] detects mental stress using a combination of ECG, GSR and accelerometer. The ECG signals were used to calculate heart rate variance (HRV) at the sampling rate of 100 Hz (same for the accelerometer) and the data from GSR was sampled at 32 Hz. Here, 20 participants (13 males) were presented with mental arithmetic problems to induce stress, and these problems would adapt the difficulty level to maintain an appropriate stress level. Moreover, the participants meditated for 10 min before solving the tasks for setting up the baseline, allowed participants to relax, and then confronted with the tasks under time pressure in three different conditions — sitting, walking and standing. A total of 10 features were extracted from HRV,

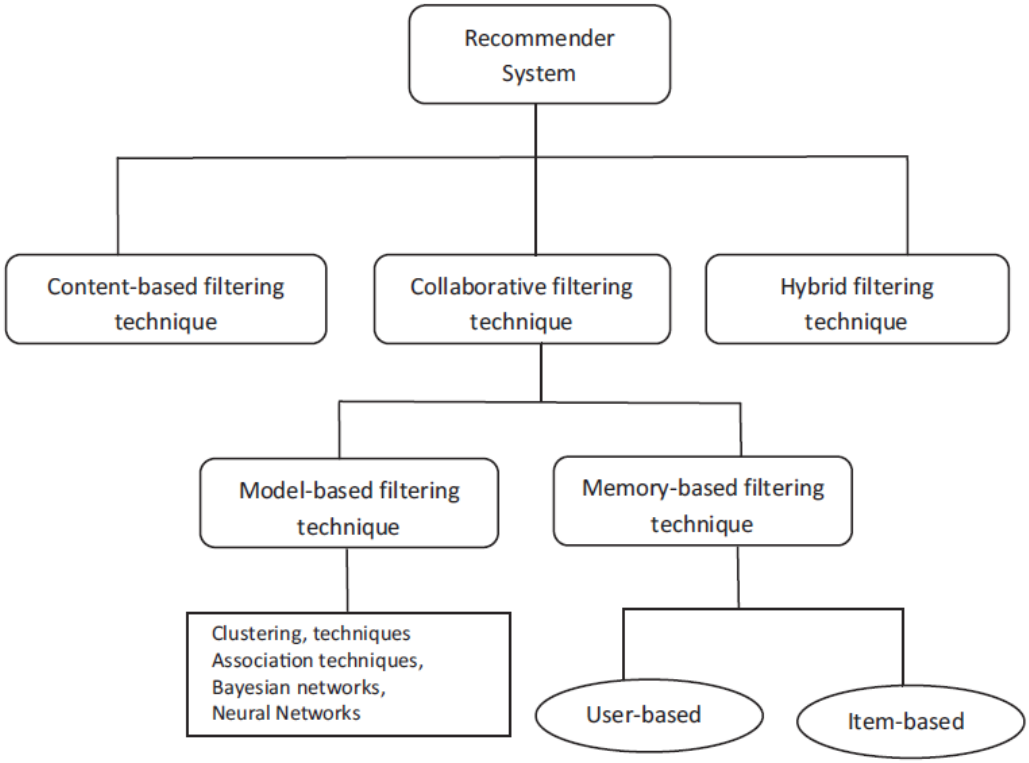

Figure 2.5: Types of recommendation techniques [18].

5 from GSR and 4 for each axis from the accelerometer. After that, the WEKA toolkit was used for testing different machine learning models and overall, decision tree achieved highest accuracy. Lastly, the authors found that the GSR features were independent of tasks.

Furthermore, research work by Afergan et al. [1] leverages functional near-infrared spectroscopy device (fNIRS) for dynamic adaption of assigning or removing the number of unmanned aerial vehicles (UAVs) to a user according to his/her cognitive status. In this, functional near-infrared spectroscopy device (fNIRS) has been to use for detecting user mental states such as a period of boredom, anxiety or overload. fNIRS uses near-infrared light to detect levels of oxygenated and deoxygenated hemoglobin in a brain. Signals from

fNIRS have been filtered to reduce noise and later fed into Support Vector Machine (SVM) machine learning algorithm for classifications. Lastly, a study by Sharma et al. [34] compared the performance of different signals and algorithms for identifying stress. They compared 13 possible physiological and physical signals and concluded that heart rate variance (HRV) outperforms every other signal for stress detection, followed by EEG then GSR.

These paper helped me to understand simultaneous working of the physiological signals, with different sampling frequency rate, and the experiment design. There are many research studies for detecting mental state using various combinations of bio-sensors [24, 26, 49] and these studies helped me to decide what bio-sensors I should use in my work.

2.2 Recommendations in Visualization Systems

An article by Isinkaye et al. [18] provides detail about principles, methods and evaluation techniques for recommendation systems. In this work, the authors talk about how the implicit and explicit feedback can be used to learn users' preferences and list the pros and cons of each feedback type. Also, the article digs deep into different recommendation techniques (Figure 2.5) — content-based filtering, collaborative filtering and hybrid filtering. Here, the content-based is a domain-dependent algorithm, and it emphasizes more on the analysis of the attributes of items in order to generate predictions, for example, web-page recommendations from keywords. Next, collaborative filtering is a domain-independent prediction technique for content that cannot easily and adequately be described by meta-data such as movies and music. Lastly, as the name suggested, the hybrid technique uses properties of both the methods for improving the quality of a recommendation. The article provided me with a clear understanding of the types of feedback and techniques that can be used to build a recommendation system. Finally, I learned and applied differ-

ent evaluation methods, for measuring the accuracy of the guiding systems, from the article.

Furthermore, Voigt et al. [43] have discussed some of the challenges of data scale and proposed a context-aware recommendation algorithm which leverages online annotations to provide help. Similarly, Gotz et al. [14] have proposed a system that generates recommendations in visualization, driven by the user behaviour (implicit signals) and successfully reduced overall task completion times and errors. Next, Hernandez-del-Olmo et al. [16] discussed the trade-off between recommendations and intrusion cost. According to the authors, the current evaluation techniques for recommender systems do not consider the intrusion cost, but each recommendation contributes to some amount of distraction.

2.3 Summary

From the past work in affective computing and adaptive systems, I gained a deep knowledge about different types of body signals that can be used to capture mental states. Also, how these signals can be processed and which type of features should be extracted for achieving high accuracy. Prior work in emotion detection helped me to understand bio-sensors devices (pros and cons) better and helped me to select appropriate sensors for my work.

Furthermore, I gained the knowledge about types of recommendation systems and their processes. Also, how I can leverage these finding for building the foundation of my proposed system.

Chapter 3

Data Collection Study

This chapter explains the data collection process and the experiment design. Since the combination of the bio-sensors was unique and there is a lack of availability of emotion datasets for visual analytic tasks, I designed a user study to collect the emotional responses from the participants while they perform the given tasks. This data was later used to train the negative emotion classifier.

3.1 Designing the Study

The first goal of this thesis was to detect negative emotions while a user performs a visual analytic task. Keeping this in mind, I had to design my study in such a way that it would induce a range of negative emotions like confusion, anger, frustration and also keep the user engaged with the tasks. For this, I followed these steps in the research process: (1) Finalized the visualization tool to use for the experiment; (2) Adopted a performance incentive approach in the experiment to engage the participants; (3) Designed the tasks; (4) Setup the hardware; (5) Pilot tested the study design; and (6) Ran the user study.

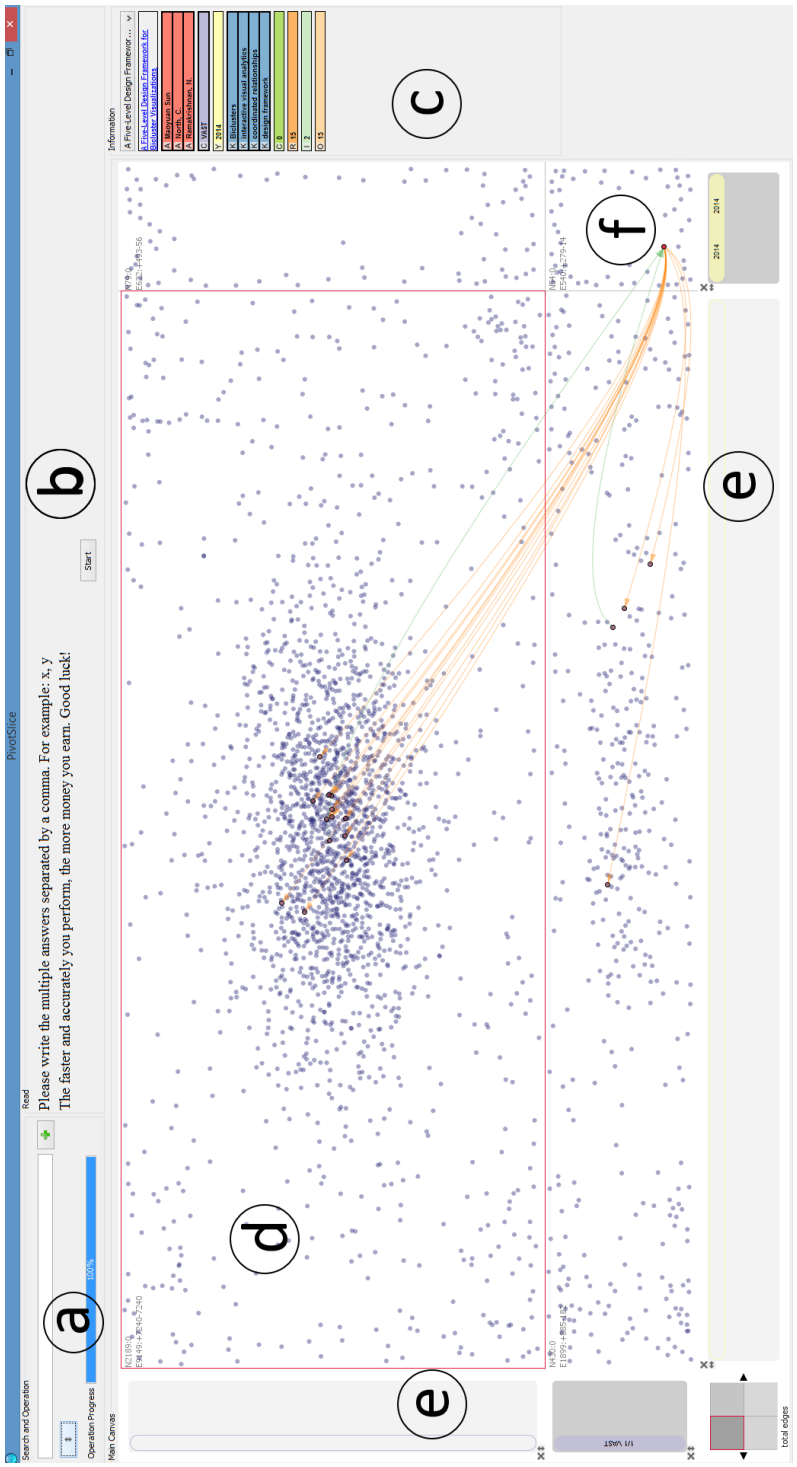

Figure 3.1: PivotSlice visualization tool [50]. The interface is divided into six sections: section *a* is search panel, *b* is task panel, *c* is information panel, *d* is the visualization view, *e* are the filter axes, and *f* is the filtered area.

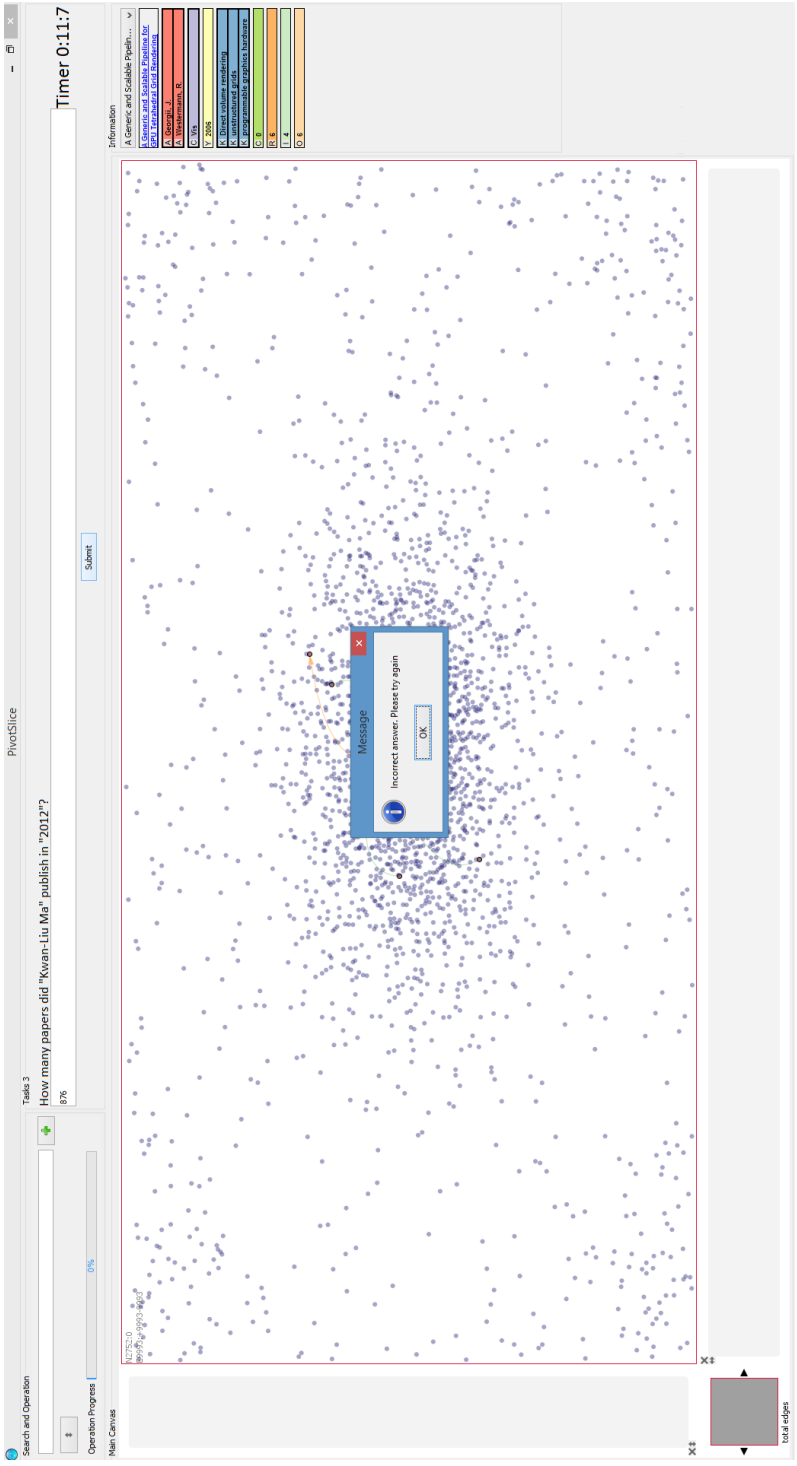

Figure 3.2: Interface popping up a message box saying "incorrect answer".

DateTime	Time (msec)	TimeStamp	User Activity	
Tue Aug 15 15:15:44 EDT 2017	15	1.50282E+12	logging starts	
Tue Aug 15 15:16:04 EDT 2017	20944	1.50282E+12	1245	1084
Tue Aug 15 15:16:04 EDT 2017	20945	1.50282E+12	1245	1084
Tue Aug 15 15:16:04 EDT 2017	20945	1.50282E+12	1245	1084
Tue Aug 15 15:16:30 EDT 2017	46889	1.50282E+12	y button	
Tue Aug 15 15:16:51 EDT 2017	67953	1.50282E+12	560	1062
Tue Aug 15 15:16:51 EDT 2017	67953	1.50282E+12	560	1062
Tue Aug 15 15:16:59 EDT 2017	75196	1.50282E+12	y button	
Tue Aug 15 15:16:59 EDT 2017	75196	1.50282E+12	1029	634
Tue Aug 15 15:16:59 EDT 2017	75198	1.50282E+12	1029	634
Tue Aug 15 15:16:59 EDT 2017	75198	1.50282E+12	1029	634
Tue Aug 15 15:17:54 EDT 2017	130448	1.50282E+12	Task1	2752
Tue Aug 15 15:19:23 EDT 2017	219820	1.50282E+12	SearchBox	/author Collins
Tue Aug 15 15:19:23 EDT 2017	219820	1.50282E+12	376	86
Tue Aug 15 15:19:23 EDT 2017	219823	1.50282E+12	376	86
Tue Aug 15 15:19:23 EDT 2017	219823	1.50282E+12	376	86
Tue Aug 15 15:19:42 EDT 2017	238017	1.50282E+12	1667	1022
Tue Aug 15 15:19:42 EDT 2017	238020	1.50282E+12	1667	1022
Tue Aug 15 15:19:42 EDT 2017	238020	1.50282E+12	1667	1022
Tue Aug 15 15:19:46 EDT 2017	242741	1.50282E+12	1655	877
Tue Aug 15 15:19:46 EDT 2017	242742	1.50282E+12	1655	877
Tue Aug 15 15:20:10 EDT 2017	267000	1.50282E+12	Task2	2012

Figure 3.3: Snippet of a log file generated by the visualization tool. Here, the user activity column records mouse click (orange), answers submitted (green), and any interface functions used by the user (blue).

3.1.1 Visualization Tool

Visualization tools allow users to interact with the data visually and the visual presentation lets users understand behaviour, trends and patterns in a big dataset. Here, PivotSilce [50] was used as a visualization tool for the user study. The interface visualizes published research documents in a scatter plot-based design and allow users to customize the visualization by adding multiple filters and making queries. For example, the total number of authors who published a journal paper in the year between 2011 and 2014 in the InfoVis conference. Also, the interface is open source, and the source code can be modified according to the experiment design.

I modified the interface according to the experiment requirements, that is, updated the database [17], removed all the extra buttons which were not useful in the experiment, included a task panel to display the tasks and added a timer which shows total time spend. The unneeded functions were removed as I wanted to minimize the possibility of getting negative emotions induced due to complex interface options and maximize the chances of inducing negative emotions due to the visual analytic tasks. Figure 3.1 shows the modified interface version with different labelled sections. Section ***a*** is a search panel where a user can apply a filter by adding a query in the search bar. The query can be a name of an author, conference, year range or a keyword. Section ***b*** is where the tasks were displayed with a timer attached (see Figure 3.2). The role of the timer was to give a sense of deadline pressure and simulate real-world scenario so that the participant will express real emotions. All the attributes of a research document such as title, author name, keyword, conference, year, reference and citations were displayed in section ***c*** which is called the info panel. These details are dynamic and change with every dot on the scatter plot, therefore, clicking on different dots will show the details of that particular research document. Section ***d*** is the scatter plot-based presentation of all the research papers. Section ***e*** is the x and y filter axis and section ***f*** is the

filtered area.

Next, only one task was shown at a time and participants were required to enter the correct answer in order to go to the next task. In case of a wrong answer, the system pops up a message box saying "Incorrect answers. Please try again" (see Figure 3.2). This strategy was used for preventing the participants to skip to the new task or end the experiment quickly without trying to solve it. Furthermore, the interface creates a log file for each session which records mouse activity (click and drag locations), interface functions that have been used and the values entered in the answer text box. Figure 3.3 shows a snippet of a log file from a session.

3.1.2 Performance Incentive Approach

For keeping the participants engaged with the tasks and induce real emotional responses closer to what a person expresses in a real-world scenario, I used the a performance incentive approach. In this, all the participants had an opportunity to earn more compensation by completing all the tasks under 20 min (threshold time). On the other hand, participants were also instructed that they could lose this bonus if they make mistakes in entering the correct answer or completing the tasks later than the threshold period. The threshold time, 20 min, was calculated in the pilot study and I developed an algorithm to compute the final compensation using the user's completion time and the number of attempts from the log file. Note that all the participants received $10 as a base amount for their participation and the approach was only applied to the bonus amount. The compensation algorithm is based on the following formula:

Step 1: *Minimum base amount that every user received*

```
baseAmount = $10
```

Step 2: *Total time is converted into minutes from milliseconds*

```
totalTime = totalTime/(60*1000)
```

Step 3: Total time is round to the closest highest integer using ceil

```
totalTime = ceil(totalTime)
```

Step 4: Updating the base amount if the total time is less than 20 minutes

```
if (20 - totalTime) >0

baseAmount = baseAmount + 2 * (20 - totalTime)
```

Step 5: Subtracting number of incorrect answers from the base amount

```
baseAmount = baseAmount - totalIncorrectAnswers
```

Step 6: If the base amount gets less than 10 then overwrite the amount

```
if (baseAmount) <10

baseAmount = $10
```

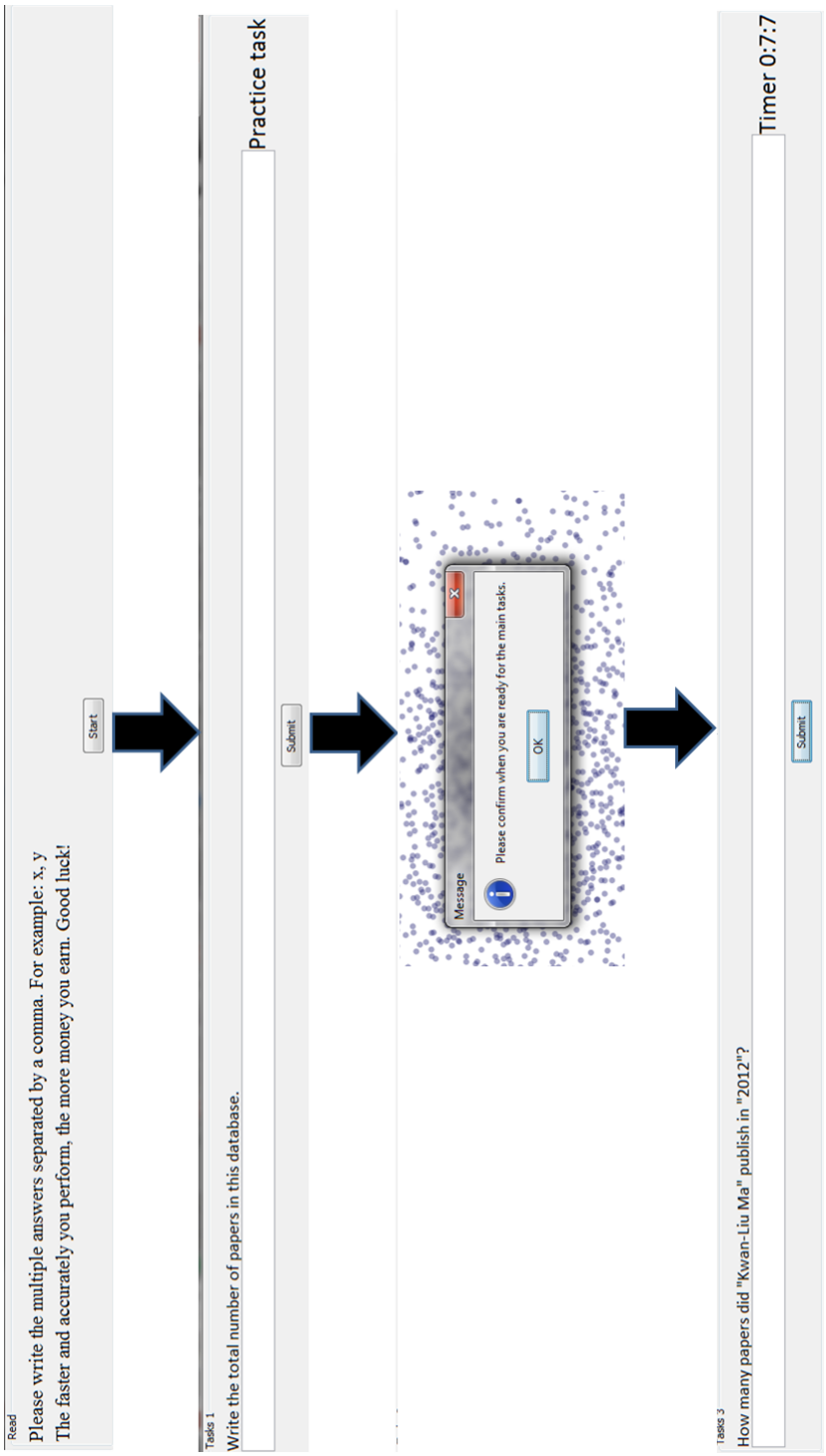

Figure 3.4: The task panel indicates the current level. The users had to click on the start button to solve the practice task. Next, the system pops up a confirmation message before moving forward to the main tasks.

For example: if a participant completes in the tasks in 17 min with 1 incorrect answer then:

```
baseAmount = $10 + 2 * (20 - 17) (Time bonus)
baseAmount = baseAmount - 1 (Incorrect answer penalty)
Final amount = $15
```

This encouraged the participants to be involved in the tasks and perform them with sincerely.

3.1.3 Task Design

I designed the data analytic tasks in increasing order of difficulty as the idea was to make participants comfortable with the interface and encourage them to solve the tasks and then gradually increase the difficulty level. The sequencing also made sure that the participants get some experience by solving some easy tasks before moving forward for the more difficult ones.

There were total six tasks for every participant — two practice followed by four main tasks. The task panel on the interface was responsible for indicating the current level — start the session, practice and main task (with the timer). Before displaying the main tasks, the interface confirms with the user by popping up a confirmation dialog box. The changes in the task panel display design for each level is shown in Figure 3.4. All the participants solved an identical set of tasks to keep the difficulty level symmetric. The practice tasks were designed to make the participants familiar with the interface functionalities and prepare them for the main tasks. Participants were encouraged to explore all the interface options during the practice tasks as their performance was not recorded at this time, only the sensors were recording their emotional responses which were independent of their performance in the task. Next, in the main tasks, the first two tasks were easy and straightforward which only

required to apply 1-2 filters and getting the output. This was again to boost up the participants' confidence level. The first two tasks are:

Task 1 - How many papers did "Kwan-Liu Ma" publish in "2012"?
Task 2 - How many papers were published in "2014-2015" with a keyword "information visualization" (case sensitive)?

The third task was considered as the medium level which required some logical reasoning and had 2 interdependent solutions:
Task 3 - How many papers were published in the "Vis" conference with a keyword "volume rendering" (case sensitive)? Also, name the author from this collection who has 5 publications?

The final task was intentionally made difficult to induce negative emotions such as confusion, anger and frustration. The final also had 2 interdependent solutions:
Task 4 - Find the paper with the highest reference count. From that paper name the author who has the highest number of publications, and the number of publications.

I tested in the pilot study, with seven expert participants, that these four tasks were enough to get a meaningful emotion dataset. Also, the designed tasks were able to induce a range of negative emotions as expected. More extended tasks would work too but the participants might feel boredom and shorter tasks would not work in the machine learning classifier due to the small amount of data.

3.1.4 Hardware

A Shimmer GSR and Tobii X60 eye tracker were used to record the user's skin conductance and gaze information at 5Hz and 60Hz sampling frequency respectively (Figure 3.5).

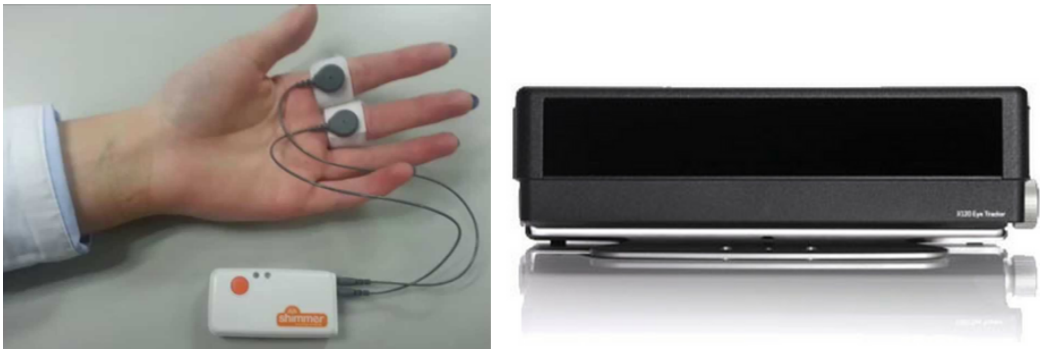

Figure 3.5: Left: Shimmer Galvanic Skin Response (GSR) device. Right: Tobii X60 Eye Tracker.

Galvanic Skin Response (GSR): Skin conductance or Galvanic skin response is a physiological measurement of the flow of electricity through the skin of an individual. When an individual is under stress or excited (aroused), the skin conductance of the body increases due to an increase in moisture on the surface of the skin, which increases the flow of the electricity [34]. The arousal can be detected by detecting a change in skin conductance.

Eye Tracker: Eye gaze movement patterns provide information on an individual's attention and enable the researcher to understand the individual's mental states and intentions [34]. Multiple attributes can be tracked using an eye tracker such as pupil dilation, blinking rate and gaze location. Pupil dilation has been examined for stress detection where an increase in pupil diameter suggests possible negative state. As pupil size is subjective to background lighting, it is common for researchers to use mean value and check the delta. It is not significant which eye gives more accurate output, but left eye is commonly used for monitoring pupil diameter [30]. Some terminologies used in this paper regarding eye tracking are *fixation* and *saccade*. Fixation is the maintaining of the visual gaze (focusing) on a single location. The saccade is the gaze movement between two fixations (distance). Again, changes in these numbers would suggest different mental state, for example — in case

of high cognitive load, users tend to fix their gaze on a certain point. This case has been confirmed multiple times in the past and again, recently studied by Fridman [11].

Furthermore, the combination of these bio-sensors was carefully selected because they were the least intrusive devices and do not require complicated setup or calibration process. The eye tracker was placed below the monitor screen, and the GSR was wearable on the wrist with two electrodes attached to fingers as shown in Figure 3.5. Moreover, these devices were less intrusive compared to EEG which needed to be worn on the head and required gel electrodes touching the scalp. Also, the EEG signals are susceptible to eye blinking or any forehead muscle movements. Other physical sensors such as motion capture or kinect or facial expression recognition were discarded due to the nature of the tasks which doesn't require too much body movement or extreme facial expressions. Additionally, the ECG device also requires multiple electrodes attached to different body parts and needs a longer calibration process to set up the baseline. On the other hand, the eye tracker only requires 10s of calibration and doesn't need to be physically attached to the user. In case of GSR, no calibration is needed and the set up was very easy.

I selected two devices as the GSR was responsible for detecting if a user is feeling some emotional changes and the eye tracker would check if those emotions are negative or not. This selection was made based on the related work section and initial investigation I did on every device. Finally, it has been proven that every device has its limitations; therefore, a combination of biosensors have shown better accuracy than using a single device in emotion detection [27, 34].

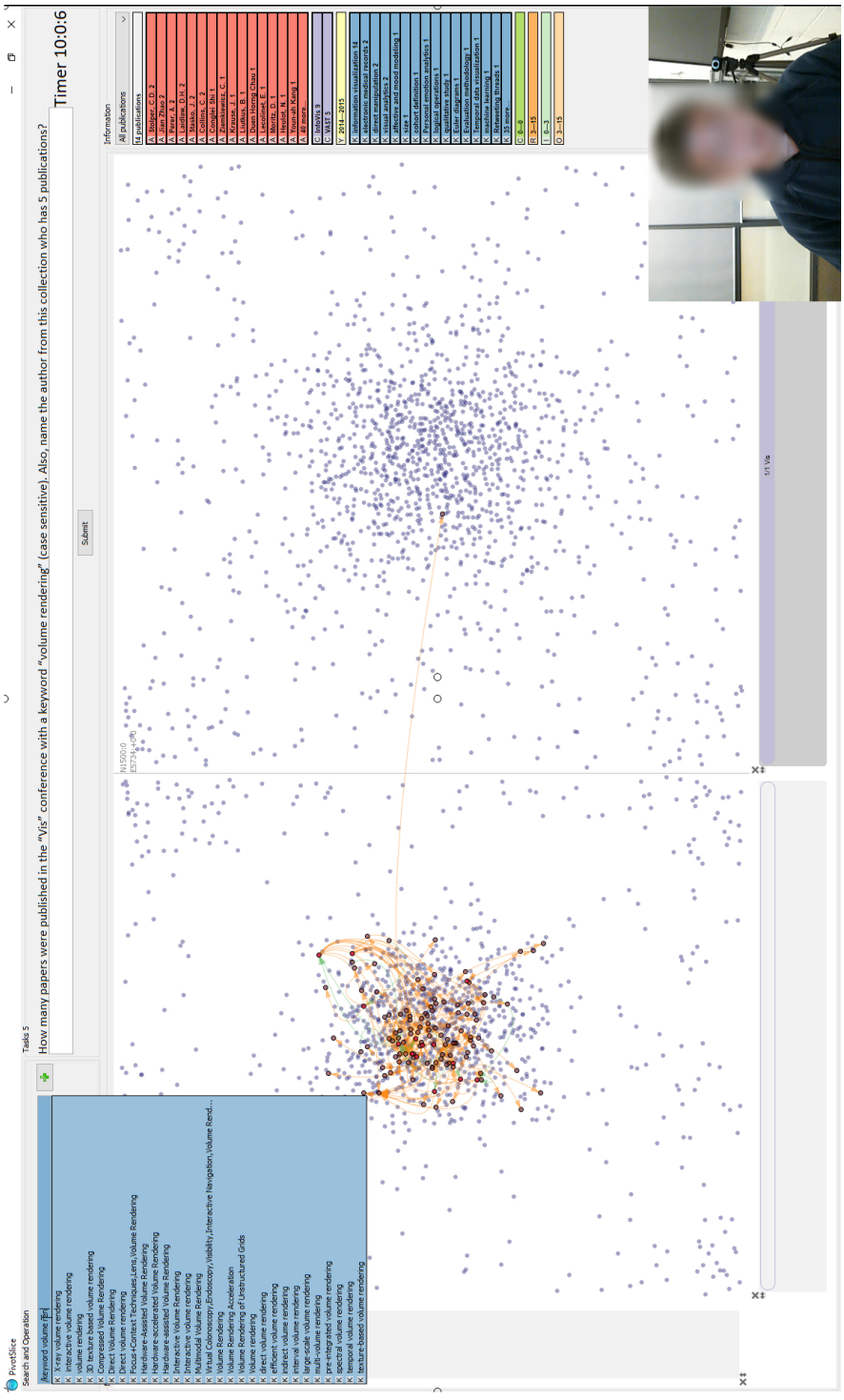

Figure 3.6: Camtasia interface showing screen and face recordings.

3.1.5 Pilot Study

After the hardware selection and the task design, I needed to make a plan for running the study which included designing the flow of the session and setting up the hardware in a quiet room where participants wouldn't get disturbed. Moreover, since there were multiple hardware involved, I needed to test my plan and fix the problems (if any) as I couldn't afford to lose any participant's data, time or money. For that, I ran a pilot study with seven expert users. These users were graduate students, Master's and Ph.D., working in a visualization lab. The pilot study allowed me to test the overall experiment process and gave me a chance to improve the design based on the suggestions from the expert users.

Experimental Setup: The room selected for the user study was specially built by the university to conduct such controlled experiments and divided into two parts — experiment room and observer room. The lighting was controlled to prevent pupil dilation due to any daylight change. Also, two surveillance cameras were used to observe the user's activity for note taking (can be seen later in Figure 3.7). For example to see if participants exhibit similar body gestures during the tasks. At the time of the task, the participants were alone in the experiment room, and I was in the observer room looking at the participant's movements with the help of the cameras. The participants were alone during the tasks to minimize the pressure or any distraction due to my presence. Also, one more camera was placed over the participant's monitor to record the frontal/facial view which was used later in the study. Finally, the frontal view from the webcam and screen was recorded using Camtasia software as shown in Figure 3.6.

Lastly, from the pilot study, I get to practice the overall session flow, made the interface more informative by adding message boxes and finally improved the data annotation part which will be explained in Section 3.2.

Figure 3.7: Participant performing the visual analytic tasks.

3.1.6 User Study

I conducted a study with 32 participants to collect the emotional responses during the designed visual analytic tasks, but due to hardware failure, the data from 4 participants were discarded. The participants were university students (26 males and 2 female), in third year or higher, from the local faculty of science. The rationale behind selecting the criteria was to choose only those users who have at least some basic knowledge of data analysis. The data collected was later used to build the recommendation system.

Session Flow: The session was divided into three parts, and each part was 20 min long which made the full session 60 min long, on an average. The participants were asked to wash their hands before the session as the GSR device is very sensitive towards sweat or any dirt on fingers. The first part was an introduction session where the participants were welcomed and were briefed about the session which included answering some questions like, why this study is been conducted, what kind of tasks they were going to perform and what data I am recording. After that, the participants were asked to sign a consent form to confirm their agreement with the experimental procedure. Next, the GSR device was set up on the participant's wrist, and the eye tracker was calibrated to record the gaze information. After the placement of the GSR device, the participants were instructed to keep the hand aside and not to move their hand with the GSR device on (see Figure 3.7). This was a crucial step to record the correct values of skin conductance as the GSR is sensitive towards movements. Due to this, the participants were restricted to use only one hand to solve the tasks which included operating the mouse and keyboard. The pilot study confirmed that interacting with the visualization tool and solving the tasks can be easily done with one hand.

After the hardware placement, some general questions were asked to the participants such as their educational background and their experience in data analysis. The motivation behind asking these questions was to make the

participants comfortable with the experiment environment and distract their attention from the hardware. Next, the PivotSlice interface was introduced to the participants, and all the basic functions of the interface were explained to them. Again, all the complex options were removed from the interface to avoid loading the participants with too much information. Also, a two-page printed interface guide was provided, and participants were told to use it anytime during the tasks in case, they forget any of the functionalities. Finally, one example was demonstrated to give them a sense of the nature of the tasks.

The second part was to performing the designed visual analytic tasks — two practice and four main. During this period, the participants were alone in the room solving the tasks, and I was in the observation room looking at the participant's activity through the surveillance cameras and writing my notes. At this moment, both the hardware were recording the body signals, the interface was recording the activities in the log file, the webcam was recording the facial front view, and the computer screen was captured. After finishing all the tasks, the interface automatically closes and records the start time and the finish time in the log file. This allowed me to trim the data recorded by the bio-sensors and use only the data which was recorded during the part 2, that is, when the tasks were performed. Part three was the think-aloud session to annotate the data manually and is explained in the next section.

3.2 Retrospective Think Aloud and Data Annotation

After the participant finished the tasks, the sensors were removed and deactivated, and all the data and recording gets saved in the system used for user study — participant's system. Now, in the next part, the data points were needed to get annotated manually for setting up the ground truth used in the machine learning classifier. For that, participants were ideally shown their screen and face recordings (Figure 3.6) and asked about how they solved the

tasks and their emotional feelings. But, given the amount of time each participant spent on first part (learning about the interface) and second part (going through a range of negative emotions) in the session, it was also learned from the pilot study that the participants feel overwhelmed, and showing them full video again and ask them recall all the emotional events on specific points was not reliable and tiring. Also, other methods such as making the participants speak aloud during the experiment or doing a retrospective think aloud after each task would cause too much interference [7]. Therefore, I needed to develop some other technique which would require a minimum amount of cognitive load from the participants and also would be accurate enough to annotate the dataset correctly.

I investigated multiple techniques and finally came up with a new approach where participants were shown exactly 7 key points in their video and asked to explain their emotional responses at that time. Additionally, I didn't ask the participants to mention a specific negative emotion at those points; instead, I just asked them to explain what they were doing at that time and how they were feeling. This way, I was making sure that the participant won't feel any pressure to say a negative emotion and hence, preventing any biased answers. For showing top 7 points, the first thing was to get a hold of participant's data as soon as they finish the tasks and then find out key moments where participants have shown signs of emotional changes.

The first step was to securely transfer the data from the participant's system to my password protected laptop where I could do all the calculation for finding the key points. Moreover, this whole process was needed to be done as soon as possible (within 1-2 min) so that it wouldn't break the session flow for the participants. To solve this, I wrote a script which closes all the data channels securely (from GSR and eye tracker) and transfers the log file, GSR data and eye tracking data to my laptop remotely using encrypted sockets. In my laptop, I wrote a Matlab script to read the transferred data and calculate the key points. Here, the key points are denoted as both positive or negative emo-

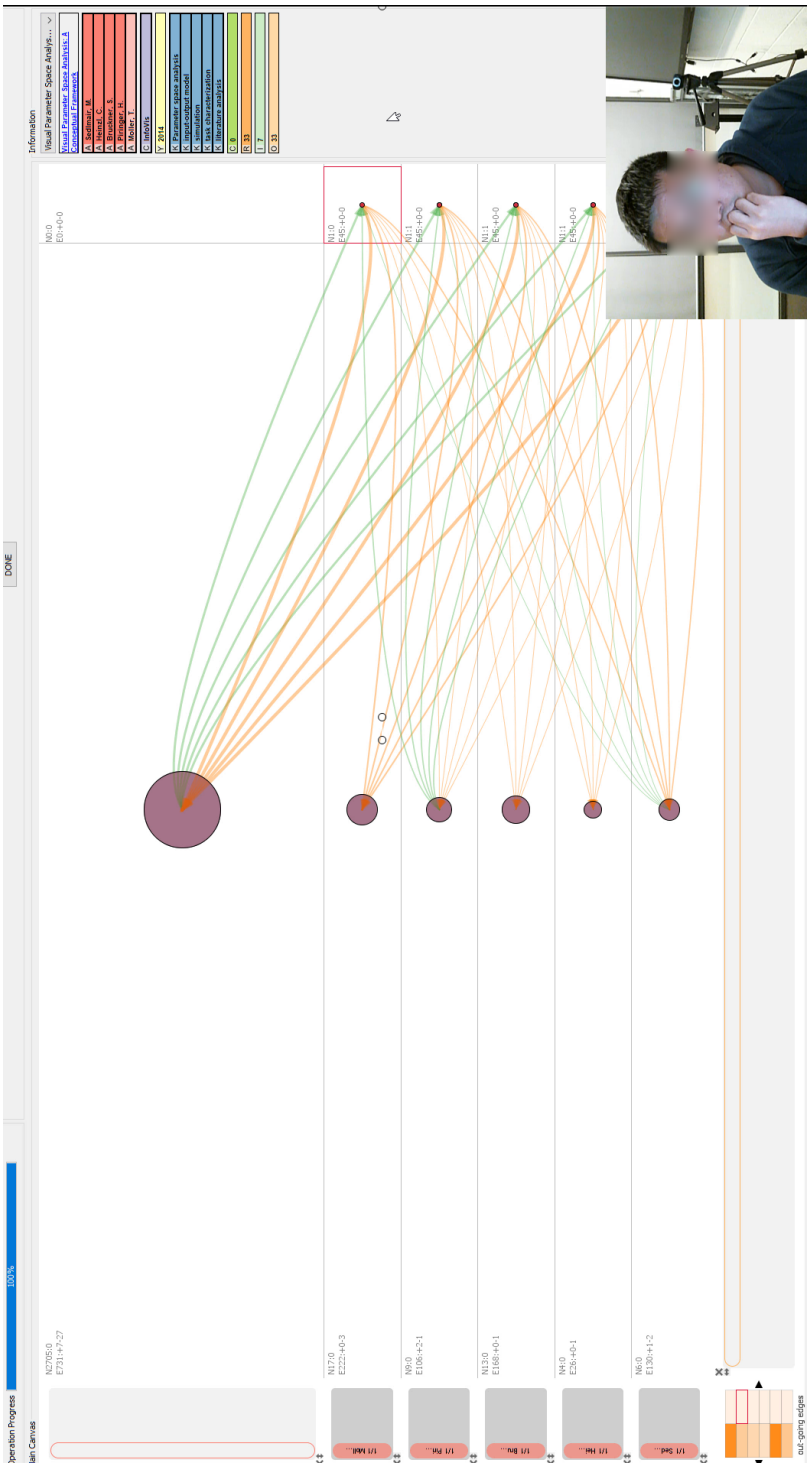

Figure 3.8: Participant is showing signs of confusion.

tions which means that the algorithm was only capable to flag an arousal state (positive or negative). Lastly, the whole transferring and calculating was done in only two mouse clicks and took an average of 60s to get key points. The details about this pre-processing and key points calculation are in the next chapter.

After I had the key points, I needed to decide how many were enough to annotate and cover the full dataset so that it could be used in the machine learning classifier. Showing all the key points to the participants would be redundant, and a lesser point would just not cover the full tasks performances. I solved this optimization problem in the user study and tested a range of different points and finally concluded to set the value to 7, that is, 7 key points were enough to cover the whole dataset. Moreover, the Matlab script calculates only those points which are at least 60s apart on the assumption that emotion would last for at least for 60s so if there are two or more key points in a 60s range that means it is likely the same emotion. Using the same assumption, I annotated all the key points within a 30s range as the same emotion. For example, if the algorithm detects a total of 30 key points and the recommended 7 point range contains 20 key points, then I am annotating those 20 key points just by showing 7 points in the video to the participants.

The 7 point strategy was faster, as the participants only had to talk about 7 points in the video which was 20 min or longer, and accurate enough to adopt for the study as in the pilot testing, the experts confirmed that they were feeling some emotional change at the 7 points shown. Figure 3.8 shows an instance where the participant looked confused, the algorithm detected this point, and the participant also verified that he was confused. I showed the video 30s before the key points to give some time to participants for recalling that moment and played the video till 30-45sec after the peak. The only limitation in using this technique was failing to annotate those instances which were flagged as emotional change but were not in the top 7. This limitation is discussed in the next chapter.

Moving on, I created another script in Matlab which uses the interface log file and calculates the final compensation amount using the formula explained in the performance incentive approach section. The participants were thanked and provided with their compensation after they finished the last part of the session.

Chapter 4

Data Processing and Classification

This chapter gives an insight on how the data, from the bio-sensors and the visualization tool log files, was processed for feature extraction and tested on various prediction models, offline. Here offline means, all the processing was done in the Matlab, and the classification part was implemented in Python. Offline testing was essential as, after the user study, I needed to perform multiple operations on the data in order to achieve higher accuracy. Then, decide how to make an online model which would predict emotions in real-time (explained in the next chapter).

4.1 Pre-processing for the Retrospective Think Aloud

As mentioned in the previous chapter, I developed an algorithm to detect key points in the participant's data. This strategy was used to make the think-aloud session faster by minimizing the time and effort of the participants.

I used GSR data to calculate arousal which is the change in skin conductance relative to the baseline. As the tasks were meant to induce negative emotion, I hypothesized that the top 7 arousal points, in the retrospective

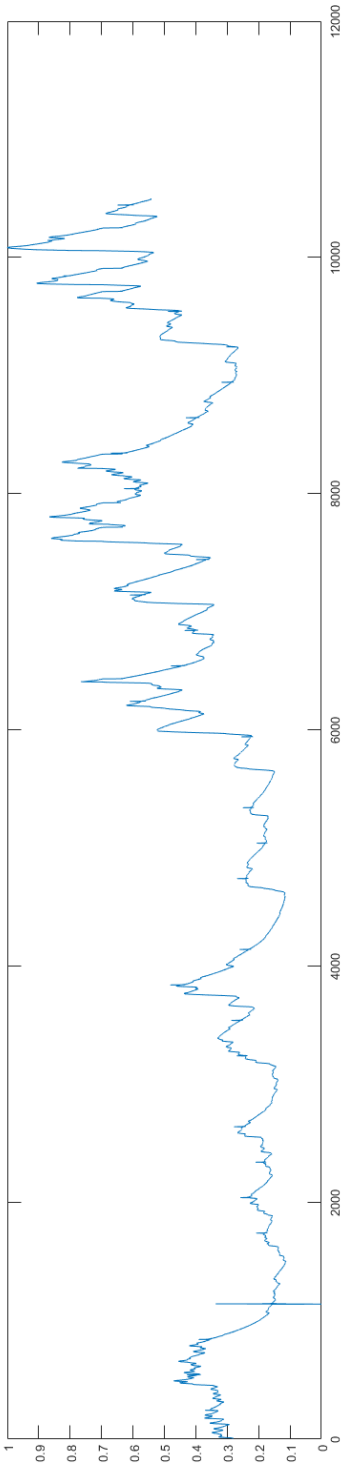

Figure 4.1: Normalized skin conductance signals in kilo Siemens (kS).

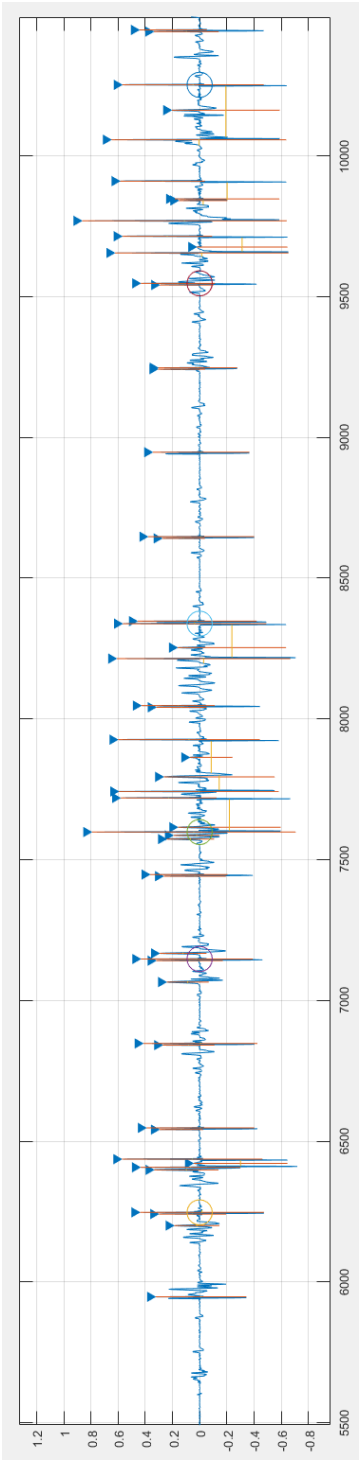

Figure 4.2: Key points detection in the skin conductance signals. The inverted triangles represents all the peaks detected and the circles are the key points detected by the algorithm. The vertical lines (red) and the horizontal lines (yellow) measures distances and are irrelevant for this Figure.

think-aloud, must have caused by negative emotions. I successfully tested this hypothesis in the pilot study by asking the participants about their emotional state at these 7 points and found that more than 97% of the time these 7 arousal points were due to negative emotions. Now, for calculating the arousal points, firstly I applied a low band-pass filter with 5Hz cutoff frequency to remove all the high-frequency noise, in the GSR signals, that occurred due to any involuntary hand movement [35]. The step was necessary in order to accurately detect the arousal. Moreover, the GSR device gives skin resistance in kOhms; therefore, I calculated the skin conductance, in kS (kilo Siemens), using the formula below:

$$\text{Skin Conductance[n]} = \frac{10^6}{\text{Skin Resistance[n]}} \tag{4.1}$$

Where n is the index in the skin resistance data. Then, the skin conductance was normalized for rescaling the data between 0 and 1:

$$SC_{\text{normalized}}[n] = \frac{SC - SC_{\text{min}}}{SC_{\text{max}} - SC_{\text{min}}} \tag{4.2}$$

Here, $SC_{\text{normalized}}[n]$ is the normalized skin conductance value at the index n, SC is the skin conductance, SC_{min} and SC_{max} are the minimum and maximum values in the data. Figure 4.1 shows the normalized skin conductance signals. Now, as the skin conductance signals are linear, I needed to find any changes in the baseline to detect arousal, and for that, I applied the 7-point second-order Lagrangian interpolation on the data [49]:

$$g''[n] = 100 \times \frac{2g[n+3]+g[n+2]-2g[n+1]-2g[n]-2g[n-1]+g[n-2]+2g[n-3]}{h^2} \tag{4.3}$$

Here, $g''[n]$ represents the second order interpolation value at index n in the data and h is the sampling frequency of the device. This gave me multiple peaks in the data, but I only selected the top 7 where the key points were at

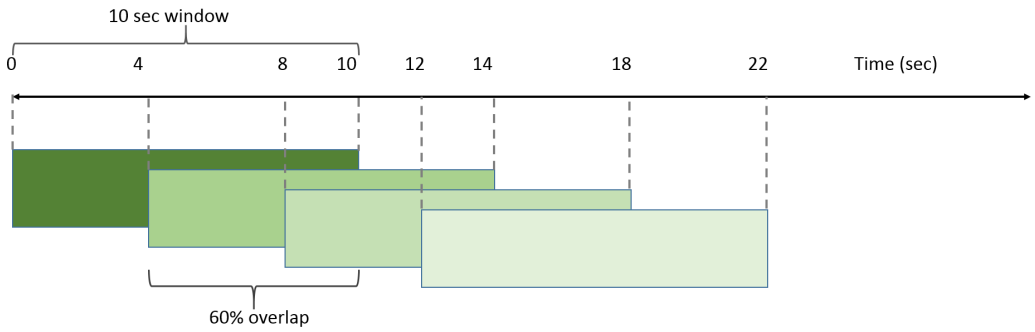

Figure 4.3: Moving window concept. The continuous stream of data was divided into a 10s window and all the operations were performed within it. After that, the window was moved to the next data block with 60% overlap.

least 60s apart. Figure 4.2 shows the snippet of the final graph — inverted triangles are all the peaks detected, and circles are the peaks which were selected by the algorithm (key points). Finally, these key points were aligned to the time when the participant started the experiment with the help of log file and displayed the exact time instances on the video where the peaks were detected. Now, as all the nearby peaks, which were 30s apart, were getting annotated, any latency at those points due to the hardware or the processing was getting resolved.

4.2 Data Processing after the User Study

This section explains all the data processing performed on the data collected from the user study. Similar to the previous section, the high-frequency noise from the GSR and pupil data were removed using a low-pass filter followed by calculation of skin conductance. Next, I used the moving window concept to divide the data into smaller blocks as the final prediction model was designed to work on a continuous stream of data, coming from the bio-sensor, in real-time. Testing the full data (offline) at once wouldn't apply in the real deploy-

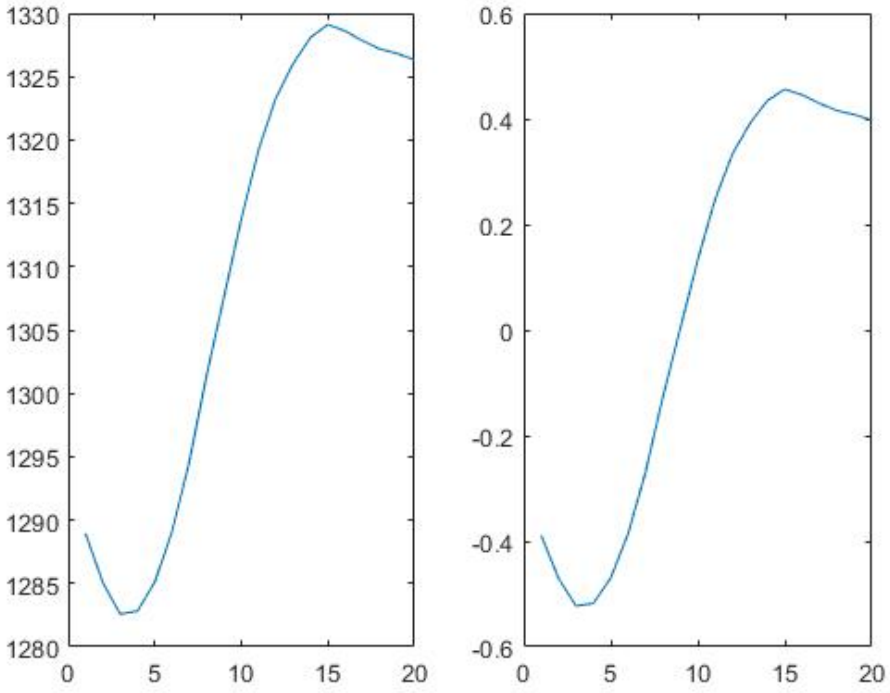

Figure 4.4: Comparison of the GSR signals behaviour. Left: Raw data; Right: Standardized data. Y axis is the amplitude of the signals in kS, and x axis represents number of samples.

ment scenario and would give wrong results; therefore I used a 10s moving window size with 60% overlap. I tested the model with different window sizes and overlaps and got the best accuracy with the settings mentioned above. A moving window of 10s contains 60 GSR samples and 600 gaze samples and to briefly conclude the concept of a moving window, I performed all the operations on only the data within 10s then I moved the window to the next 10s (with an overlap of 6s) and then performed the same calculation. This allowed me to work on a continuous data. Figure 4.3 explains the working of moving windows with time.

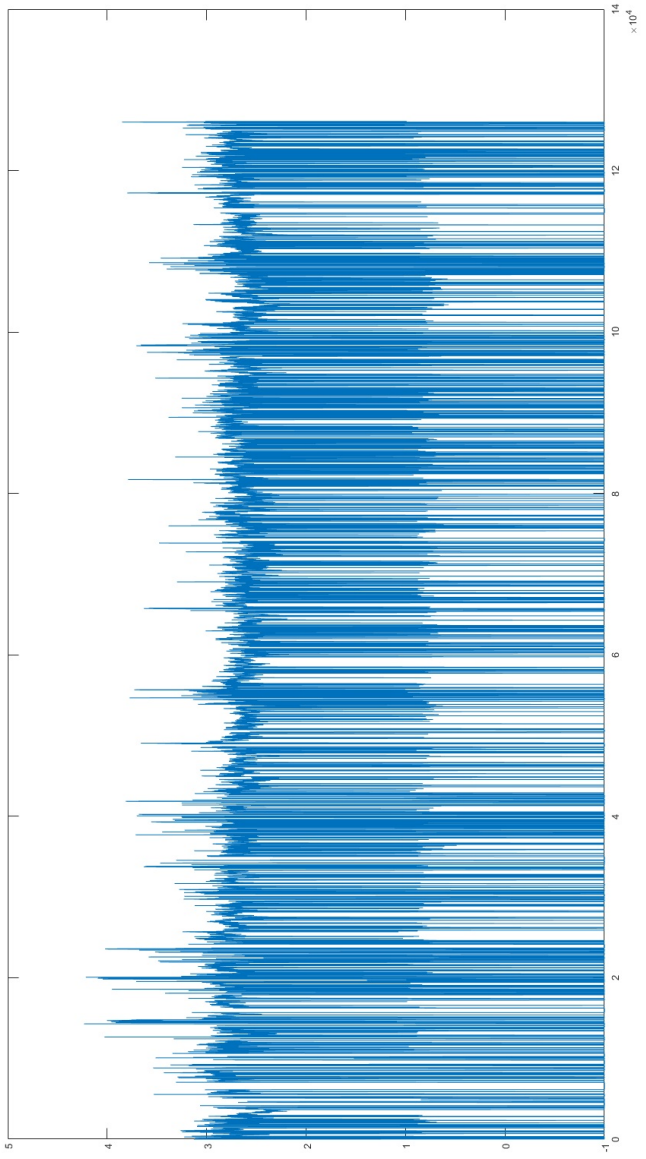

Figure 4.5: Raw pupil diameter signals in millimetres. Here, the values less than 0 represents cases where the eye tracker was unable to track the pupils, for example when the user was blinking.

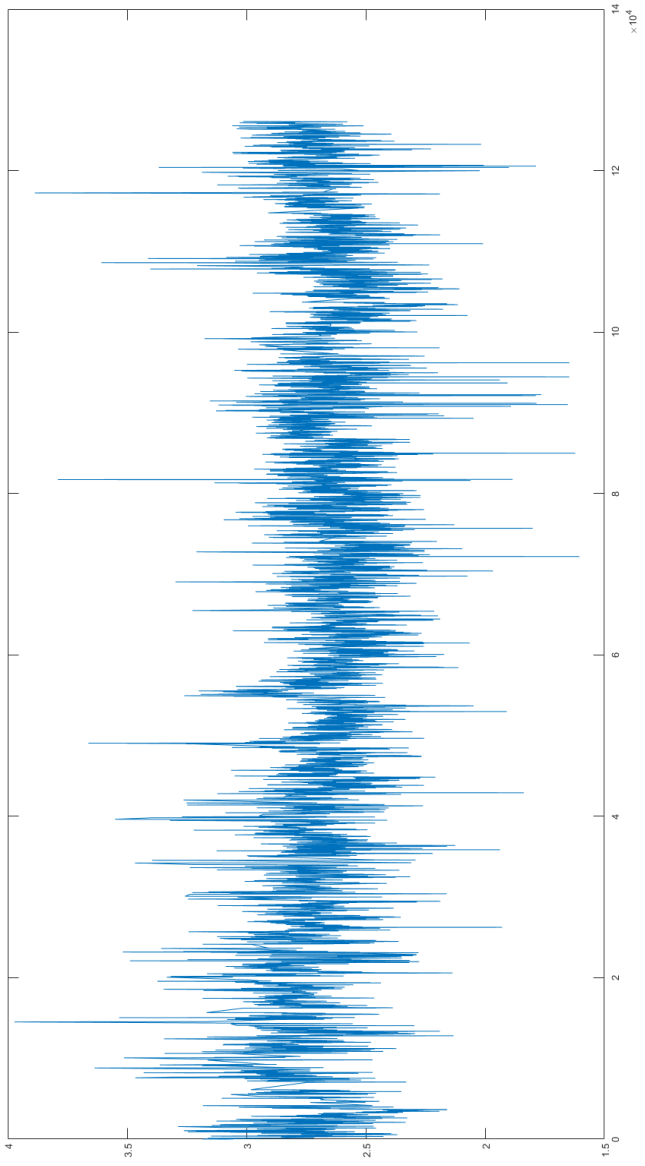

Figure 4.6: Processed pupil signals. Samples with the low confidence were replaced by interpolation and high frequency noises were removed.

Furthermore, raw pupil size data and GSR data were not generalizable as every person exhibits a different range of emotions with different baselines and therefore, cannot be used for training a machine learning classifier. For that, I used standardization method by calculating the z-score to scale the baseline without disturbing the nature of the data (Figure 4.4).

$$\text{zscore} = \frac{x - \mu}{\sigma} \tag{4.4}$$

Here, x is the raw value in the data set, μ is mean, and σ is the standard deviation of the data. This technique is an alternative of normalization and best work for a moving window because, in normalization, the minimum and maximum values vary for every window. In case of z score, I calculated mean and standard deviation from the first window till the $n\,t\,h$ window for keeping the values consistent and comparable between each window. Figure 4.5 shows raw pupil data with noise and Figure 4.6 shows processed and filtered pupil data.

The next part was to filter the gaze data — gaze locations on the screen and blinking. In case of blinking or too much head movement, any eye tracker fails to record accurate data and these samples were needed to filter out in order to generate the final dataset. For that, the Tobii eye tracker SDK calculates a confidence score for each sample from -1 to 4 where -1 mean no eye detected and 4 is highest tracking confidence. Also, the documentation provided by the Tobii suggested to only consider samples with a confidence score of either 3 or 4. I followed the documentation and removed all the samples with 2 or lower score. Before deleting the samples with lower confidence, I counted the number of -1s and saved the count to use it later. These -1s could point out some interesting patterns in the participant's behaviour. Details are discussed in the later section. Finally, after saving the -1s count, I removed the samples with low confidence, and I applied linear interpolation technique to fill those deleted values.

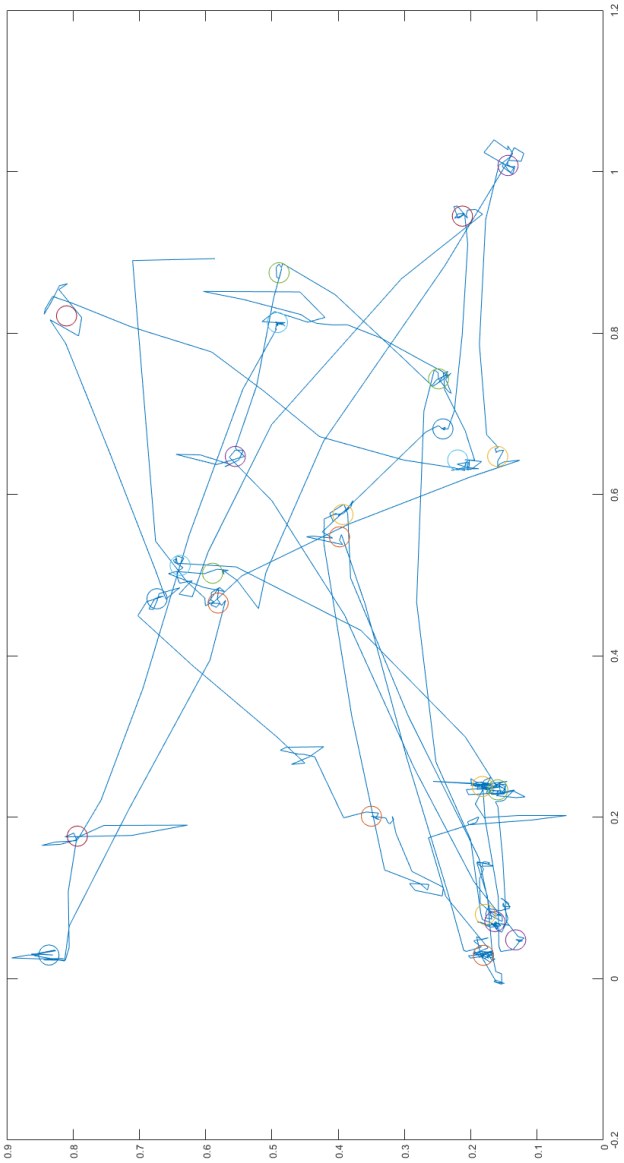

Figure 4.7: Fixation and saccade patterns from participant 14 gaze data. The gaze path is represented by blue color and the circles are the fixations detected. These fixations were calculated using equations 4.5, 4.6 and 4.7.

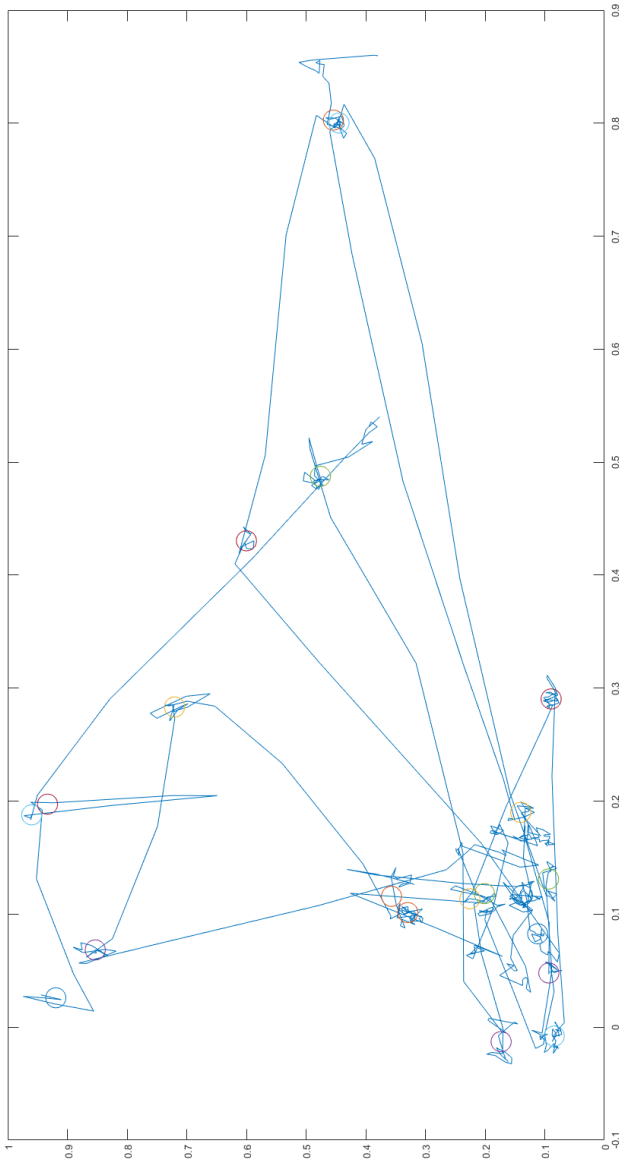

Figure 4.8: Fixation and saccade patterns from participant 21 gaze data. Fixations are noted with circles along the blue gaze path. These fixations were calculated using equations 4.5, 4.6 and 4.7.

Furthermore, I used gaze location information to calculate the focus points, called fixations, and the distance between two successive fixations, called saccades. Any sudden variations in fixations and saccades might indicate an event. For example, a large number of saccades and fewer fixations may indicate visual searching related to confusion and frustration or more fixation in the same area and fewer saccades could be an indication of high cognitive load. I used the method illustrated by Olsson [29] to calculate fixations and saccades between two consecutive inner sliding windows.

As gaze movements are swift and the eye tracker is recording the samples at 60Hz, calculation of fixations and saccades had to be done on smaller windows (than the main 10s window) to detect gaze activities more accurately. For example, if a gaze is travelled 50 times between point A and B in 20s but 40 times was under first 5s then, analyzing this behaviour in a bigger window would be overseen. Therefore, a smaller window size is needed to catch these behaviours. For that, I created a 5s nested window with 4s overlap inside the main 10s window to get a better insight into the gaze patterns. Moreover, Olsson suggested that taking too small a window would increase the chances of counting noise as fixation and taking too big window size would count two nearby fixation points as 1. These nested sliding windows helped in breaking down the continuous data in the 10s main window into more smaller blocks.

For calculating the fixations and saccades, the overall idea is to take a mean of x and y coordinates of the gaze location data of two consecutive inner windows and then calculate the distance between them. Once this step is done on the full 10s gaze data, apply a threshold to select larger distances which are saccades. Then, finding the median between two saccades will give fixations. Following are the steps to calculate saccade:

$$m_{before}(n) = \left[\frac{1}{r} \sum_{k=1}^{r} s_x(n-k), \frac{1}{r} \sum_{k=1}^{r} s_y(n-k) \right] \tag{4.5}$$

$$m_{after}(n) = \left[\frac{1}{r} \sum_{k=1}^{r} s_x(n+k), \frac{1}{r} \sum_{k=1}^{r} s_y(n+k) \right] \tag{4.6}$$

Where s_x represents the gaze location in x axis and s_y is gaze location in y axis in a window where n is the index of the sample of interest, and r is the inner window size in seconds (smaller window inside the main 10s window). Olsson illustrated this method by taking 4s as the value of r, but I tested different window sizes — 4s, 5s and 6s, and finalized 5s based on the accuracy I obtained. For calculating the means between two windows, the distance was calculated as:

$$d(n) = \sqrt{(m_{\text{after}}(n) - m_{\text{before}}(n)) \cdot (m_{\text{after}}(n) - m_{\text{before}}(n))^T} \tag{4.7}$$

Where d is a distance vector of length $N-1$ and N is the number of inner sliding windows. Out of all the distances, I only selected those which were more than the standard deviation of the vectors. These selected distances were marked as the saccades. The fixations were calculated by finding the median of the samples between two consecutive saccades. Figure 4.7 and 4.8 are the visual representation of gaze paths from two different users in a 10s data window. Circles are the fixations detects on the raw gaze path represented by the blue line.

These are all the steps in the data processing. The processed data then fed into a function to calculate the features which is explained in the next section.

4.3 Feature Extraction

Uniqueness in feature selection is what makes a classifier perform better, therefore choosing the features is one of the most crucial steps in building a machine learning model. For that, I did a thorough study of the related research and finalized the features which would contribute the most and are sensitive

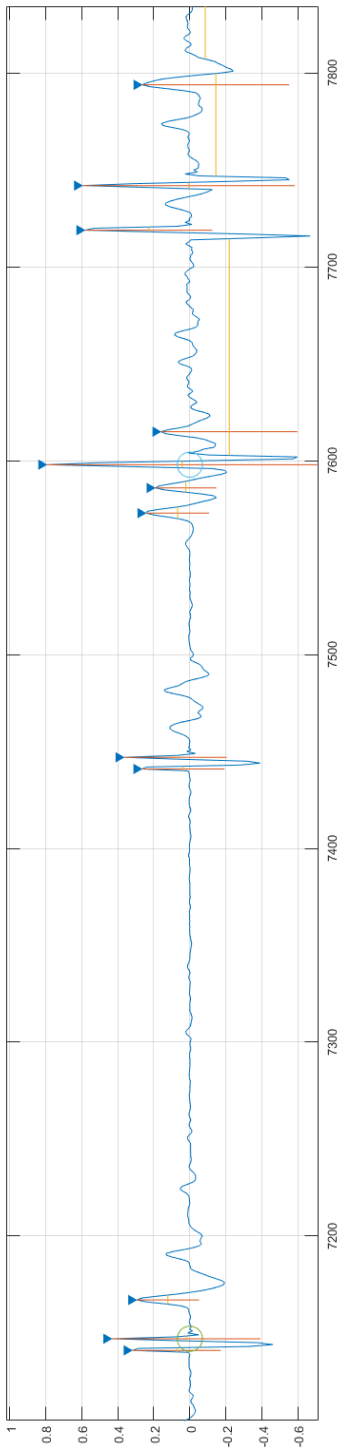

Figure 4.9: Processed GSR signals after applying the 7-point second-order Lagrangian interpolation using the equation 4.8. Here, inverted triangles represent all peaks detected and circles are the peaks selected. The vertical lines (red) and the horizontal lines (yellow) measures distances and are irrelevant for this Figure.

towards any changes in the baseline.

To train a classifier model, I calculated a total of 21 features from the processed dataset — 8 from the pupil signals, 8 from the GSR data and 5 from the gaze location data.

Standard deviation, mean, number of peaks and sum of the height of the peaks were calculated from GSR and pupil size data. These signals are linear and exhibit similar behaviour; therefore any change in the baseline would affect those features. Next, I applied 7-point second-order Lagrangian interpolation (also mentioned in the data processing section) on the standardized pupil and GSR data and calculated the same above mentioned features again:

$$g''[n] = 100 \times \frac{2g[n+3]+g[n+2]-2g[n+1]-2g[n]-2g[n-1]+g[n-2]+2g[n-3]}{h^2} \quad (4.8)$$

This formula was validated for finding good peaks in the retrospective think aloud, and therefore, applying the same approach for feature extraction would make sense. The 7-point second-order Lagrangian interpolation is used for finding the points where linear signals changed relative to the baseline [49]. Figure 4.9 shows a part of GSR data which was processed, and the 7-point second-order Lagrangian interpolation was applied. In this Figure, the signals were re-scaled using standardization (z-score) and are much smoother as the high-frequency noise was removed. The peaks show changes in the signals where higher the amplitude means more significant change.

Finally, five features were extracted from fixations and saccades, calculated in the data processing section, to detect any change in the gaze pattern — number of fixation, mean saccade length, mean fixation duration and standard deviation of the fixation points from the centroid of fixations in the window (indicating the extent of the area of interest). Again, these features were selected because they are sensitive towards attention and cognitive load, for

Signals	Technique	Features
GSR	z-score	Mean Standard Deviation Number of Peaks Sum of Height of the Peaks
GSR	Lagrangian	Mean Standard Deviation Number of Peaks Sum of Height of the Peaks
Pupil	z-score	Mean Standard Deviation Number of Peaks Sum of Height of the Peaks
Pupil	Lagrangian	Mean Standard Deviation Number of Peaks Sum of Height of the Peaks
Gaze Location	Fixation & Saccade	Number of Fixations Mean Saccade Length Mean Fixation Duration Extent of Area of Interest Blinking

Table 4.1: List of calculated features from GSR and Gaze data.

example, a user tends to focus on a smaller area in high cognitive load [11] and therefore, the number of fixation reduces with smaller mean saccade length. The final feature was the total number of -1s which indicated the blinks (explained and calculated in the previous section) or when the user was not looking at the screen. A high value of this feature might indicate mind wandering or boredom.

Table 4.1 list all the features that were used for the classifier with respect to the signals and techniques applied. The classification models are discussed in the later section.

4.4 Classification Model

After the feature extraction, I calculated the ratio of the events (negative emotion) and non-events class in the final dataset which came out to be 1:12 as most of the time participants were not exhibiting strong negative emotions. This highly imbalanced data or also know as *rare event detection problem* in machine learning wouldn't work on any classification model. There are two ways to resolve an unbalanced class issue — oversampling and undersampling. Oversampling is to generate artificial data of the minority class, and undersampling is to remove data from majority class. Here, I used the oversampling technique as the dataset was not large enough and the cost was high to ignore any of the cases from the sample size. To perform an oversampling technique, the Synthetic Minority Over-sampling Technique (SMOTE) [5] was applied to balance the classes. This algorithm uses the k-nearest neighbour based method to generate artificial data points along the line of minority data. By default, it uses five nearest neighbours and randomly pick a point on the minority data clusters which allows the algorithm to generate artificial data which has similar behaviours as of its neighbours. Furthermore, to prevent bias in the accuracy of the classifier due to oversampling, SMOTE algorithm

was only applied to the training data, and I kept the original unbalanced data for testing. This allowed me to eliminate those cases in the testing which were artificially generated by the SMOTE algorithm and might boost up the results by over-fitting.

Furthermore, after resolving the unbalanced classes case, I tested the dataset on various supervised machine learning classifiers. For that, I used the *sci-kit learn* library in Python and then tested the dataset on several algorithms: support vector machines (SVM), k-nearest neighbours (KNN) and random forest. Also, I used the *Tensorflow* library to test the dataset on neural networks but due to the smaller size of the dataset, all the classifier types such as simple neural network, Deep neural network and Recurrent neural network; had failed to achieve more than 30% of accuracy. I used 28 fold cross-validation (hold out one participant at a time) to test the accuracy of different models. Random forest with 1000 trees and max depth 4 outperformed other models and gave 88% of recall score. Since the test data was imbalanced, rather than using a simple accuracy (which would be high even for a naive classifier), I report confusion matrix and negative class recall score. The recall score tells accuracy of a model in differentiating between events from the non-events. For example, if there are 10 negative events and 120 normal states in the dataset and a classifier couldn't detect any negative events, it would still give more than 90% accuracy based on a large number of non-negative events, but the recall score for the negative class would be 0. To calculate the recall score, first, I generated a confusion matrix of the predicted result and then used the following recall score formula:

$$\text{Confusion Matrix} = \begin{bmatrix} \text{True Negative} & \text{False Positive} \\ \text{False Negative} & \text{True Positive} \end{bmatrix} \tag{4.9}$$

$$\text{Recall Score} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \tag{4.10}$$

Models	Recall Score
K-Nearest Neighbors	55%
Support Vector Machine	67%
Random Forest	88%

Table 4.2: Recall of the negative class with different classification models.

$$\begin{bmatrix} 171 & 25 \\ 1 & 44 \end{bmatrix} \quad \begin{bmatrix} 137 & 45 \\ 0 & 28 \end{bmatrix}$$
$$\begin{bmatrix} 199 & 39 \\ 2 & 25 \end{bmatrix} \quad \begin{bmatrix} 236 & 16 \\ 2 & 20 \end{bmatrix}$$

Table 4.3: Confusion matrices of random forest classification from four different participants. 1st column represents non-event and 2nd column is negative response.

Here, *True Negative* represents a state which was labelled and detected as a normal state. Similarly, *True Positive* is the state which was labelled and detected as a negative emotion. On the other hand, *False Positive* was labelled as normal but detected as a negative emotion, and *False Negative* was labelled as negative emotion but detected as a normal state. Since one of the primary goals of this work was to provide help when a user needs guidance, calculating and comparing the recall score of the negative emotion class made more sense. Table 4.2 shows accuracy, in terms of negative class recall score, of all the models tested on the emotion dataset.

Moving on, Table 4.3 shows four confusion matrices from four different participants which gave an insight into the random forest classifier accuracy. Using the first example from the table, 171 are the non-event (true negative), 25 are false positive, 1 is false negative and 44 represent negative emotions

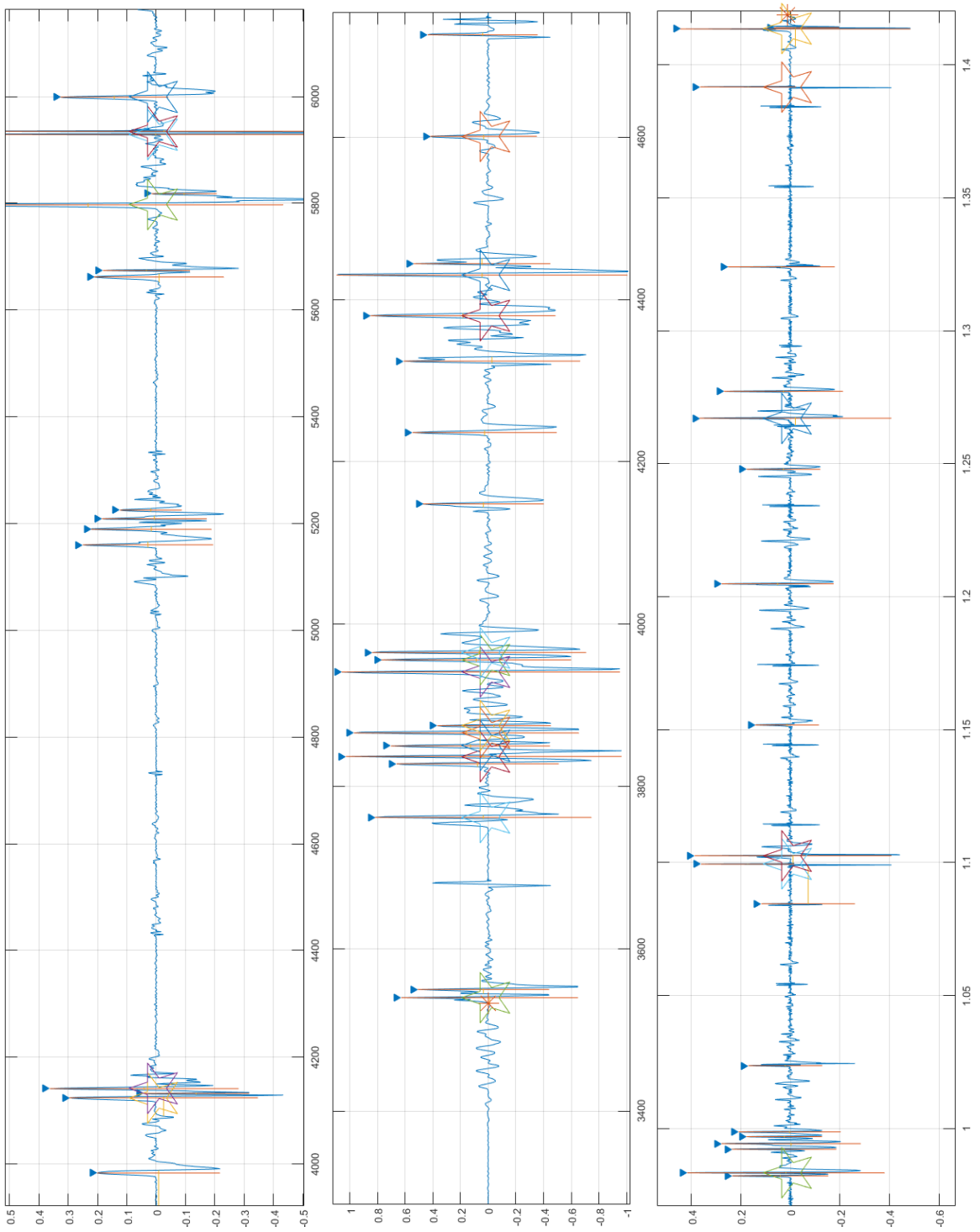

Figure 4.10: Comparing the similarities between labelled peaks (marked as starts) from various users' data and the peaks which were not labelled. Here, the inverted triangle are all the peaks that were detected by the algorithm.

(true positive). The model detected high false positive which means that some events that were labelled as a normal state are being detected as emotion events. Two primary reason could cause this — either the model was not appropriately trained, or it was detecting the negative emotions that were missed in training data as they were not in the top seven key points discussed in the think-aloud session. Labelling the dataset based on seven key points has its advantages, but there was always this risk that some of the key points would be missed in labelling. I hypothesized that the model was working correctly and the cause of high false positives was due to the missing labels. Moreover, if the model was not predicting correctly or was over-fitting, then it would not give a high recall score and low false negative.

In addition, Figure 4.10 represents GSR signals from different users where stars were marked on the peaks that were labelled as emotional change. In this Figure, it can be noticed that several peaks exhibit similar behaviour but were not labelled as an emotional change due to the algorithm rules — selecting only seven top points which were at least 60s apart and labelling similar peaks within 30s of that point. Therefore, this might indicate that the classifier also flags those emotion states which were not labelled in the dataset. As the final goal of this work was to provide a variety of types of assistance, providing unwanted help (false positive) is less problematic, if designed well, than missing potential moments of frustration and confusion (false negative). In a nutshell, using recall score again helped me to justify the correctness of the model and gave the actual accuracy by differentiating between true positive events (negative emotion) and true negative events.

Signals	Technique	Features	Labels
Pupil	z-score	Mean	a
		Standard Deviation	b
		Number of Peaks	c
		Sum of Height of the Peaks	d
Pupil	Lagrangian	Mean	e
		Standard Deviation	f
		Number of Peaks	g
		Sum of Height of the Peaks	h
Gaze Location	Fixation & Saccade	Number of Fixations	i
		Mean Saccade Length	j
		Mean Fixation Duration	k
		Extent of Area of Interest	l
		Blinking	u
GSR	z-score	Mean	m
		Standard Deviation	n
		Number of Peaks	o
		Sum of Height of the Peaks	p
GSR	Lagrangian	Mean	q
		Standard Deviation	r
		Number of Peaks	s
		Sum of Height of the Peaks	t

Table 4.4: Labels used for each feature.

Steps	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u
Start	.44	.30	.24	.20	.37	.16	.37	.35	.52	.18	.48	.19	.70	.62	.65	.71	.43	.81	.68	.51	.55
r	.83	.79	.78	.80	.83	.78	.76	.81	.78	.79	.85	.78	.81	.84	.83	.83	.81		.80	.82	.83
rk	.79	.80	.81	.80	.83	.81	.77	.78	.82	.79		.79	.77	.82	.80	.80	.82		.82	.80	.81
rke	.78	.83	.80	.81		.84	.80	.80	.81	.83		.83	.81	.83	.82	.81	.83		.83	.79	.82
rkef	.78	.82	.81	.82			.80	.80	.80	.81		.79	.79	.84	.80	.82	.80		.83	.79	.80
rkefn	.80	.84	.81	.85			.83	.80	.84	.83		.82	.81		.83	.83	.82		.84	.82	.82
rkefnd	.81	.83	.83				.83	.80	.83	.82		.82	.81		.83	.81	.88		.81	.80	.81
rkefndk	.81	.84	.79				.83	.78	.83	.82		.81	.80		.81	.83			.82	.82	.80
rkefndkb	.83		.84				.83	.81	.82	.82		.81	.82		.83	.80			.81	.81	.83
rkefndkbc	.82						.82	.81	.82	.83		.81	.81		.81	.81			.82	.79	.81

Table 4.5: Feature combinations and accuracy.

Ranking	Features
1	Std GSR (Lagrangian)
2	Mean fixation duration
3	Mean PD (Lagrangian)
4	Std PD (Lagrangian)
5	Std GSR (zscore)
6	No. of peaks PD (zscore)
7	Mean GSR (Lagrangian)

Table 4.6: Ranking of the finalized features.

4.5 Code Optimization

After building a working negative emotion classifier, the final step was to make the model more efficient by reducing the feature vector dimensionality. This step was required because the goal of this thesis was to get feedback from users on-the-fly and decide a suitable recommendation in real time. For optimizing the model, I calculated and selected only those features, from the current feature set, which were contributing the most in achieving the highest accuracy. For that, I used brute force technique to calculate the ranking of each feature based on the recall score and tested the model with all the possible arrangements and combinations of the features. Table 4.4 shows the naming convention (labels) used for each features in Table 4.5.

As can be seen in Table 4.5, step 1 (`Start`) is where I tested the accuracy of single features. The feature with the highest accuracy is highlighted as green, in the table. In later steps, I merged the selected feature combinations from the previous step with new features and calculated the accuracy. As can be observed in the table, step 7 (`rkefnd`) reached the highest accuracy (0.88), and following combinations gave accuracy approximately consistent accuracy between 0.83-0.84 then starts to drop in later steps (not in the table). Once I noticed this pattern and the maxima where the accuracy was highest, I selected

that features combination and dropped all the remaining features. The final seven features helped the code to run faster as the dimensionality of the feature set was reduced from 21 to 7. Table 4.6 shows the list of finalized features and their ranking.

In summary, to this point, I collected the emotional response data from user study then did data processing, extracted features, tested the classifier, and optimized the model to work online. Next, this thesis will talk about how I made the model online and how the system generates the recommendations.

Chapter 5

Real-Time Prediction

This chapter starts the second half of this thesis. As the prediction model was able to differentiate between negative emotions and non-negative states with a recall of 88%, the next step was to make the model online so that it would provide the prediction in real time. That is, receiving the data from two different bio-sensors (GSR and eye tracker) at different sampling frequencies, sync the signals in a moving window, process the signals, extract the features, feed it into the machine learning model and finally get the predicted output. The initial model wouldn't work online as the bio-sensors APIs were in C#, the preprocessing was happening in Matlab, and the prediction was built in Python.

5.1 Receiving the Data

The first step was to receive the data from the bio-sensors instead of saving it in a file so that the prediction model could work on a continues stream of data. The problem with this was that the bio-sensors APIs were in C# and the prediction model was in Python. Moreover, receiving data from two different devices required to run two different C# programs in parallel and then send

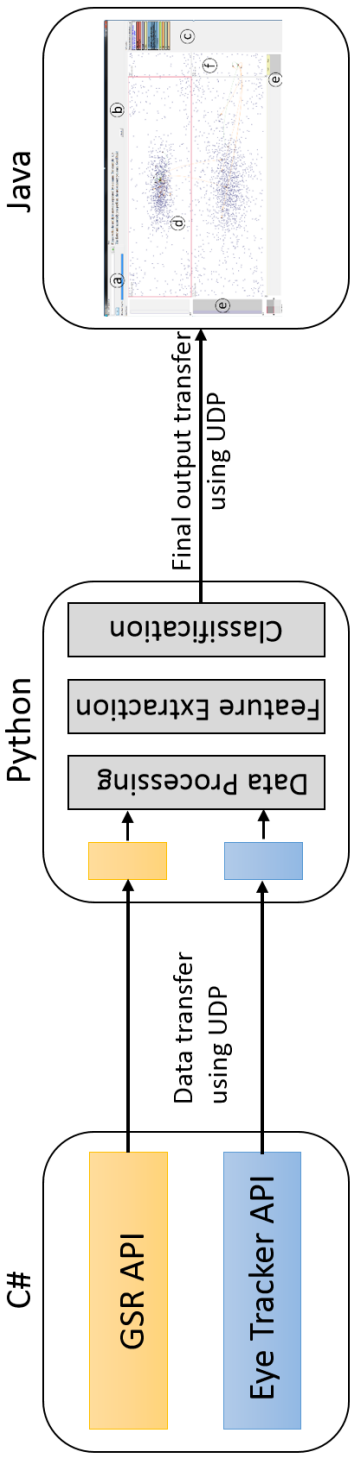

Figure 5.1: Block diagram of the model. The communication is happening in real-time between different languages using UDP transfer.

those data to the Python function. To solve this, I used socket transfer approach which allowed me to build a communication path between the Python and the C# programs and to minimize data loss or any latency while transferring the data. I used User Datagram Protocol (UDP) method as the data transfer was happening in the same system and there was no need to add additional security offered by Transmission Control Protocol (TCP). The overall idea was to send data packets from the bio-sensor APIs (as clients) to the Python function (as the server). Moreover, I assigned the clients two different ports with same local IP to prevent any data conflicts and made the server to listen to the assigned ports. Figure 5.1 illustrates the internal communication of the system. Now next issue was to sync the data received and is discussed in the next section.

5.2 Merging and Parallel Processing

Since the data from two different bio-sensors was coming in parallel with two different sampling frequencies (GSR at 5Hz and eye tracker at 60Hz), therefore, I needed to receive it in parallel so that I could work on a data stream recorded at the same time. For that, I used the multi-threading approach in Python and assigned two separate threads that were responsible for extracting data from the server (listening on two different ports) and feed it into separate the data processing functions.

After successfully receiving the data in parallel, in real time, the next part was to merge the data processing function, feature extraction function and the classification model; and for that, I re-wrote all these parts in Python. Since the operation I applied in the data processing was independent of any Matlab core functionalities, it was more comfortable for me to transfer all the concepts to Python using the open source data processing libraries such as *Pandas* and *Numpy*.

Ranking	Features
1	Std GSR (Lagrangian)
2	Mean fixation duration
3	Mean PD (Lagrangian)
4	Std PD (Lagrangian)
5	Std GSR (zscore)
6	No. of peaks PD (zscore)
7	Mean GSR (Lagrangian)

Table 5.1: Ranking of the finalized features.

Moreover, now as the code will do the processing in real time, I needed to sync the data coming from GSR and the eye tracker in a 10s window for further processing. Firstly, the two threads that were receiving the data from the bio-sensors calls two different functions for filling the data into two separate queues and these samples were synced using timestamps. I wrote a separate function that compares the timestamp to verify that data in the queues are in sync or not. If the timestamps don't match then this function raises a flag and stops the program. This extra step was necessary to avoid predicting on wrongly synced data.

Next, these queues were sent to two different data processing functions so that whole data can be processed in real time. Similar to processing Chapter 4, all the high-frequency noise gets removed from the signals using a low pass filter, and the signals get standardized with the z-score calculation. After that, low confidence gaze samples were replaced with the interpolated data and fixations and saccades were getting calculated from that data. Here, all the steps were similar to the offline testing which was done in Matlab.

Next, in the feature extraction part, the same seven features were calculated (Table 5.1), in parallel, from the processed data saved in two separate queues — GSR signal queue and gaze signal queue. After the feature extrac-

tion part, I needed to merge these features in order to feed them together in the classifier. For that, I added a function which checks if all the seven features are updated or not. If not, then it keeps on checking and hold calling the classification model function; if all the features are updated, then it calls the prediction model function and passes these features as an argument to the function. This function helps in keeping the features in sync because in a case where one thread is running faster than the other thread then there is a chance where old values from the slower thread will be merged with the newer values. For example, if the GSR thread finished calculating the features earlier than the gaze thread or vice-versa, then the function would merge the new GSR features with old gaze features. Therefore, I implemented this intermediate function which was responsible for keeping the process in sync and calling the classification model only when both the threads finished processing.

To summarize, I implemented two intermediate verification functions — (1) before the data processing; (2) before the classification. As the data is coming in real time with different sampling frequencies and all the processing is done in the real-time too using the multi-threading approach; therefore, there was need to insert these verification processes to avoid any *syncing* mistakes. Moreover, this step also makes my system robust.

Furthermore, before testing the real-time negative emotion detection system, the final step was to skip re-training of the model every time I start the process. For that, I trained the model separately and saved it using *Pickle* library in Python. Then, in the primary function, I called the saved model before starting the threads and used it for predictions.

Lastly, I examined the working of the model on an $i5$ intel processor with 10 Gb RAM and checked for the latency. For that, I simulated the data receiving part with the saved data from the participants which means that instead of receiving data from the bio-senors, I wrote a function that reads participants'

raw data and feed it to the threads in precisely same frequency rate. Considering the bio-sensors API and the main Python function was independent, it wouldn't interfere or change the processing time of the primary function. Moreover, this approach allowed me to compare the predicted results of the real-time model with the offline model.

Now, with a 10s window and 60% overlap, the model was supposed to be predicting in every 4s, ideally. After structuring, the final negative emotion detection model was classifying in between every 4–4.5s with a maximum delay of 0.5s and because, there was a 4s gap between each detection, 0.5s (worst case) didn't matter. Moreover, the detected results were approximately similar to the offline model with ±3% difference.

5.3 Sending the Predicted Output

Till now, I created a real-time negative emotion detection model which gives a prediction in every 4–4.5s. The next task was to sending these classified results back to the visualization tool (in real-time), PivotSlice, for further processing of generating a recommendation. For that, I again used UDP socket programming approach for sending the data live from the Python program to PivotSlice which is built Java. In the backend, I created a separate class in Java for receiving the classified output and for further calculations. Again, this class uses multi-threading technique to prevent any interference in the operations done on the front end of the visualization interface.

Finally, this finishes creating the model on-the-fly and sending the detected negative emotion back to the visualization tool. The next chapter will discuss the process of analyzing these results and generating the recommendations.

Chapter 6

Intervention

This chapter describes how I used the detected negative emotions to generate meaningful recommendations. For that, let's revisit the introduction chapter first where I discussed the three essential steps — "when, what and how"; to provide a recommendation. First, deciding *when* a user needs help and then triggering the recommendation function. Next, determining the source of the negative emotion (context) and then deciding *what* type of help is needed. Finally, deciding *how* to provide the help by balancing the intrusion level. It is crucial for the system to determine an intervention based on the intensity of the detected negative emotion, where intensity represents the length of the negative emotion. For example, glowing a help icon when the intensity is low and increase the intrusion by popping up help when the intensity is high. In the later section, I have discussed a method for calculating the intensity of the negative emotion. Also, in this chapter, I explain in detail that how each of the questions mentioned above can be solved using the classified emotion and gaze location information as feedback to the system.

After all the signal processing and machine learning classification, I am sending the final prediction to the PivotSlice interface as — `T` for true (negative emotion) and `F` for false (normal state). Here, interface backend is receiving

these outputs using the UDP socket programming approach and computing "when, what and how" to intervene.

Additionally, observing these negative emotions for an individual user or in a group could reveal some interesting trends in the visual analytic task. For example, analyzing emotions of an individual for different visual analytic tasks could give an insight of user's problem-solving skills such as performance or common areas where the user felt similar emotion. Similarly, there is a potential of analyzing emotions of a group for a specific task or an interface for determining the flow and use it for building a model which doesn't need to record emotions.

I have demonstrated some of the ideas on a specific interface, but these ideas are generalizable for other visualization interfaces. Moreover, in this chapter, I have also explored the design space of the interventions and recommendations that could potentially aid in the improvement of the users' performance. The methods are explained in detail but only a few ideas were implemented, and none of them were tested with the users due to time constraints.

6.1 Detecting *When* to Help

Every 4s, the visualization system is receiving the detected user's current emotional state in real-time. Also, saving these outputs in a list and monitoring the sequence could potentially reflect the duration of emotion which could likely indicate the intensity of emotion or could be useful in predicting user's learning curve. For example, if the system has received constant T for more than 15 times (60s), then the negative emotional intensity is high compared to receiving 7 Ts in 60s.

Since the system is classifying user's emotional state in every 4s, the next

task was to trigger the suggestion function only when the system confidence is high, rather than taking action every 4s. This is because taking actions every 4s would make the system too sensitive, would consume more computational power and also could be tedious or annoying. Additionally, calculating some confidence score would make the system robust towards wrong classifications. Let's think of the case where the system detected negative emotion, and it is popping help in every 4s which diverts the analyst from his task and inducing more negative emotion which system is reading again. For preventing this loop, there was a need to smooth the transition and trigger an action only when the system is confident which can be determined by a rule-based or a machine learning technique. Here, I used a rule-based method.

6.1.1 Action Transition Smoothing

Instead of taking action every 4s, I created a moving window of 32s (8 classification in a window) to smooth the action transition and build up confidence. The rationale behind choosing a 32s window was to provide enough data to take a reasonable and meaningful action but also minimizing the risk of mixing two separate emotions in a single window. Due to time constraints, I did not investigate the optimum window size for this case and saved this part for the future work.

The system is counting total Ts in the 32s window, and if total Ts are more than a threshold value of 5 Ts (20 out of 32s), an action will be triggered. The threshold is an average value based on the predicted output patterns observed in the participants' data. Also, taking a higher threshold would make the system less sensitive and avoid taking actions on those short-term negative emotions which lasts for less than 20s. This threshold could easily be tuned in further testing, or even dynamically changed in real time. For example, if the user repeatedly dismisses recommendations, the threshold may go up.

If the total T are 5 or more in the window, an action is triggered. After that, the system sleeps for next 60s before monitoring the next window. This period allows users to look at the recommendation, use it and if it helped then alleviate. Since negative emotions do not come back to the normal state right away, therefore, the system should ignore the classification right after a recommendation has been displayed. This would also prevent the system to take repetitive actions and avoiding a loop discussed above. The latency in triggering an action after the onset of a frustration event is dependent on the order of detection as well as the window size of 32s. Actions are only triggered after the window is filled with the detected outputs. The best case scenario is 20s after the start of frustration and the worst case scenario is 32s. The following examples demonstrate these cases:

F F F T T T T T — Best case scenario
T F F F T T T T — Worst case scenario

Here, the first sequence contains five constant Ts after three Fs. As soon as the moving window counts the first T (at 4th place), it would take 20s to reach to the final T (at last place) and then an action will be triggered. In the second sequence, the moving window has to come till the end (8th place and 32s after) after counting the first T (at 1st place) in order to trigger an action. Since the goal was to detect long-term frustration states, this latency would not affect the efficiency of the system. Also, the threshold value, which is 5 here, could be reduced to adjust this latency.

6.1.2 Intensity of an Emotion

Another thing is to calculate the intensity of an emotion. As mentioned above, calculating the intensity of detected negative emotion can be used to infer the intrusion level. Here, a different sequence of T and F can tell the intensity

Sequence	Weight Distribution
T T T T F F F F	1+2+3+4 = 10
T F T F T F T F	1+1+1+1 = 4
F T T T T T T T	1+2+3+4+5+6+7 = 28

Table 6.1: Weight distribution of sequences. Calculating intensity of negative emotion.

of the negative emotion or might also differentiate between types of negative emotion. For example, following are the three cases where a sequence of T and F are different in a window:

F T T T T T T T
T T T T F F F F
T F T F T F T F

In the first sequence, there is a constant occurrence of Ts after the first F. This might be an indication that the user is feeling strong negative emotion. Next, the number of Ts and Fs are same in the last two sequences, but the ordering is different. The first sequence has four constant T which means a high-intensity negative emotion (short term) which faded afterwards and the second sequence means light but consistent negative emotion. Since these two negative emotions are different, therefore the help for these sequences should be different too. Also, the variation in these two sequences might classify the different type of negative emotion and could be explored in the future work.

To differentiate between the sequences, I have created a rule and assigned non-uniform weights to T and calculated total weight for each window. The rule is — any T after an F would have a weight 1 and any immediate T after a T will have a weight one more than the previous T. Table 6.1 shows some sequences and their weight distributions. By using this formula, I was able to differentiate between various sequences and hence calculated the intensity of

an emotion. Thus, now the system can condition the type of help based on the intensity of an emotion.

6.2 Deciding *What* to Recommend

As mentioned in the introduction chapter, generating a meaningful recommendation for helping users mainly relies on knowing what is making them feel a negative emotion. In other words, knowing the context is important here because a user can feel same emotion for different reasons. Based on the user study results and the prediction model's capability, I classified three different contexts which are first identified by the proposed model and then contextual help is generated. In visual analytic tasks, a user can feel negative emotion due to *the interface, dataset, or total disengagement*. Knowing the context would help in maximizing the chance of generating a useful recommendation. For example, if a user is having problems with the new visualization interface, then the system should not show any dataset-related help for improving the understanding of the dataset. The help should be related to the interface options or else the generated suggestion won't be able to help the user to overcome the issue and negative emotion. To differentiate between each state, the system uses gaze location information and calculates centroid point of the area of interest using fixations made in a particular window. Also, adding more variable such as the location of the most extended fixation and comparing the fixation time for deciding the context would be more robust but was not implemented in this work.

Moving on, the classification was happening in a 10s window (600 gaze samples) therefore, it was likely that the user's gaze path would be more inclined towards the source of negative emotion on the inference. Also, observing the sequence in the 32s emotion window would help the system to decide better. For example, if the system detects that a user is feeling a strong nega-

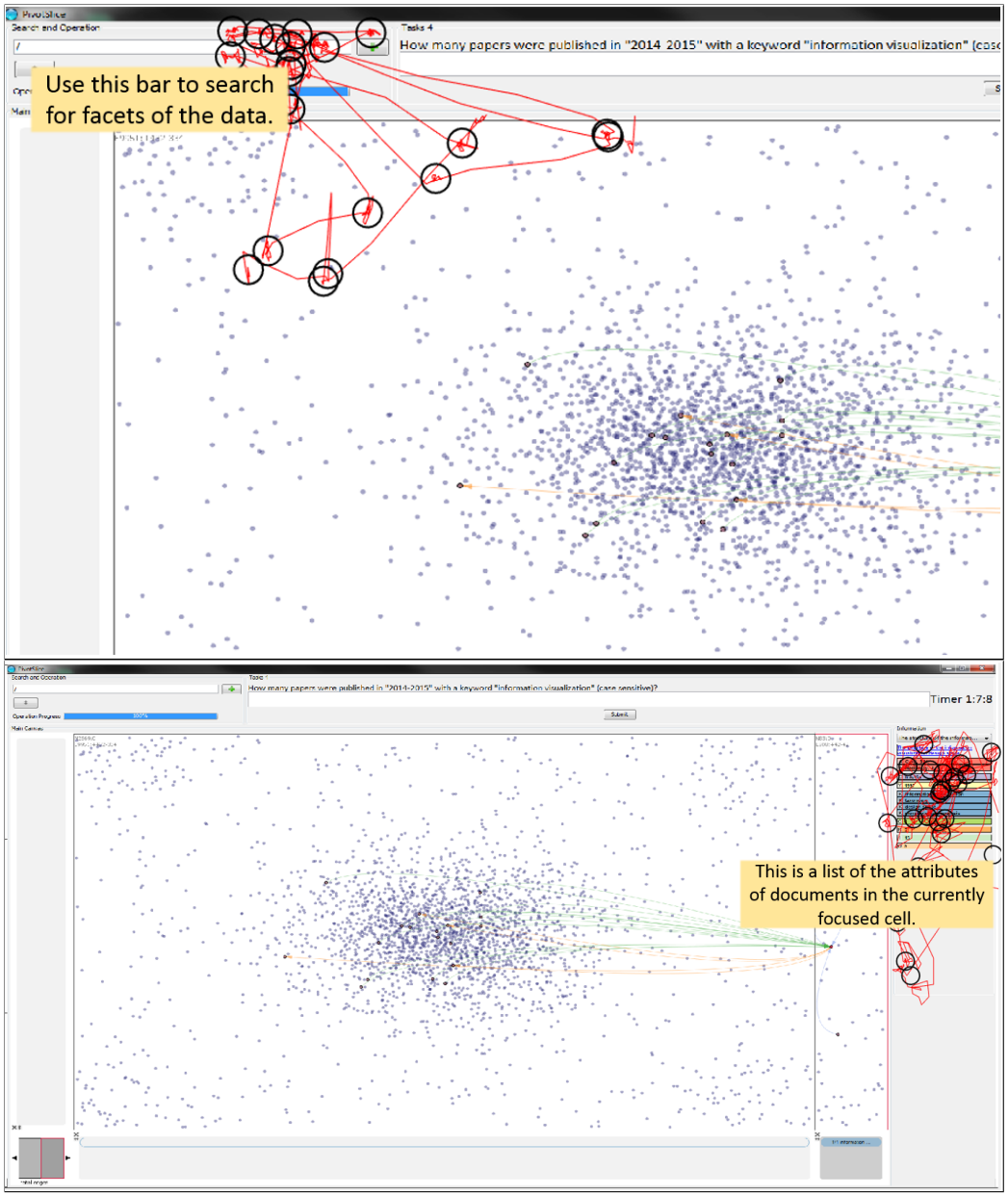

Figure 6.1: The system is using gaze location information to determine where the user was looking at the time of negative emotion. Later, the system is providing a recommendation based on that area. Top: Search panel related help, and bottom: Info panel related help.

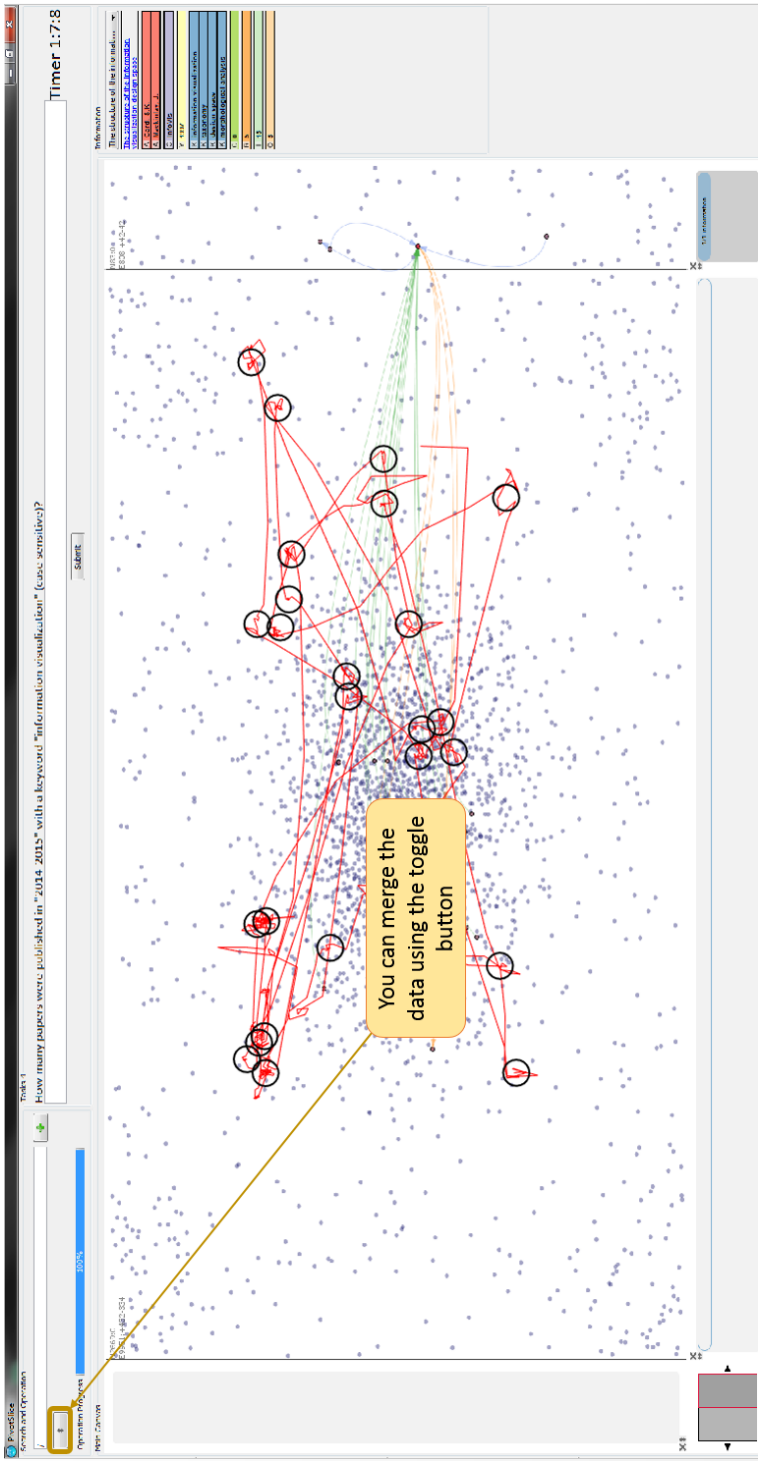

Figure 6.2: Dataset operation suggestion. Here, the system detected the context as dataset-related and then suggesting an operations that can be done on the dataset.

tive emotion and out of eight times (total samples in the 32s emotion window), five times the centroid was on the interface toolbar, then it implies that the user might be having problems using the interface functionalities. Therefore, a centroid point would decide what part of the interface made the user feel a negative emotion. Figure 6.1 demonstrates system capabilities, with help of an example, in finding the possible source of negative emotion with the help of gaze location. In the Figure, the system is detecting the area on the interface when a negative emotion was detected and displaying a dialog box. The design dimensions of a recommendation, such as where to show the message box and what color to use, are not studied in this work. The three context classes are discussed in detail below:

6.2.1 Interface

An interface-related problem occurs when a user doesn't understand the interface functions and options. This type of problem may be more likely to occur with a new interface or one with many functions on the screen. Also, in the user study, all the participants were new to the interface, and even though detailed instructions were demonstrated to them in the introductory session, they used the interface manual often. Since going through the manual every time is redundant, consumes time and a visual search task by itself, the design of recommendations for the interface is thus focused on providing contextual help on the screen to reduce the need to consult the interface manual (one of the possible solutions is shown in Figure 6.1). This would prevent the users to look away from the screen and hence would be an efficient way to sustain the attention.

Figure 6.3: Participants disengaged and looking away from the task screen

6.2.2 Dataset

Dataset related problem occurs when dealing with a new, unfamiliar dataset or a substantial multi-dimensional dataset. Here, showing interface-related help is not meaningful. Instead, showing different operations that are possible on the dataset could be one of the recommendations. Figure 6.2 demonstrates a mock example where the system is suggesting that dataset can be merged to get common information and also highlighting the button on the interface for merging. Other ideas are, showing interesting parts in the dataset [47], data tours, suggesting the user change the visualization view or highlighting hidden and unseen points in the dataset. Moreover, a step-by-step guidance is also possible to help the user to carry out complicated operations.

6.2.3 Disengagement

Disengagement is the case when a user is not looking at the screen. In the user study, I have observed that when participants felt frustrated or confused for an extended period, they tend to look away from the screen (Figure 6.3). Using voice dialogues can be helpful in re-orienting the user's attention back to the screen as done by D'Mello et al. [9]. That being said, in the real world scenario, there are other factors could make the user look away and forcing a user to look at the screen could be annoying. Again, it is difficult to say if the voice-based recommendation would work in this cases or not. Therefore, there is need to investigate this case in detail for suggesting better recommendations, and I did not implement this case as it would require a separate study to understand and identify different causes for disengagement. Hence, for this work, the system can detect disengagement by analyzing the gaze data (loss of eye tracking data for an extended period), but does not take any action.

6.3 Deciding *How* to Recommend

Finally, after computing when to intervene and what to recommend based on different contexts, the next and the final step is to show the help. Again, it is crucial to display the suggestion in a way that it would not distract the user from the task and would adjust the intrusion level according to the intensity of the negative emotion. Deciding the way of showing a recommendation and considering the intrusion level are the two major components discussed in this section. A study [16], also mentioned in the related work chapter, talks about the importance of considering the intrusion level in recommendation systems because useful help can still be annoying if displayed in an intrusive way.

For better understanding of the intrusion level, let's consider an example in the PivotSlice interface. If a user doesn't know how to apply two or more filters then the system can help in three possible ways: (1) Giving a hint by displaying a short message over the search panel; (2) Break down the whole process and guide the user by showing the instructions step-by-step; (3) Pause the interface and open the instruction manual. All three ways can help the user, but the question is which one is the best? One of the possible answers can be to show help (1) when the emotional intensity is low, and the user needs help for the first time. Help (2) is useful when intensity is medium, and the user didn't understand help (1). Help (3) is the extreme case where the user is continuously getting confused with the filters functionality and therefore a detailed understanding is needed, before going further. If the system shows the same suggestion every time, then it won't benefit the user and possibly would overwhelmed the user leading to disengagement.

6.3.1 Degree of Intervention and Guidance

Since I have calculated the intensity of the detected negative emotion in the previous part using weights, I am using that to calculate how much to intervene. The overall idea is, when negative emotion is detected, the system will analyze it and compute how much to intervene based on the user's actions. For example, displaying a simple and less intrusive help when the intensity of the emotion (weight) is low such as, highlighting options or displaying hints about dataset on the side. Now, for adjusting the intrusion level, I illustrated a simple approach — every suggestion can be tracked by the interface logging to see if the users followed the recommendation or not. The system can alter the guidance strategy based on this information. For example, if the suggestion says to use the search bar for creating a filter, then the backend part of the system can track if the search box was successfully used or not, after the help.

If the user has used the suggestion and the window weight (negative emotion intensity) is less than the previous window that means the help worked. If the user has used the suggestion and still the total weight of the window is increasing or constant, that could mean the user needs more guidance, and the intrusion level can be increased. Furthermore, if the system detects that a user is feeling negative but not using the suggestion, that might be an indication the user doesn't need that help. In this case, the system will display a new help and keep the degree of intervention same.

In conclusion, this was the rule-based approach that I explored in term of *how* a recommendation system should balance the intrusion level and what factors a system should consider facilitating a suggestion. Again, replacing this method with a machine learning approach may make the decision making more effective, but this would require more data.

6.3.2 Exploring Ways to Generate Recommendations

After discussing how to show a recommendation and what are the factors that should be considered before intervening, the next step is to how to generate a recommendation that would guide the users and helps them to sustain the engagement. For this, I investigated three possible scenarios and I explain below how a recommendation could be generated based on each scenario.

When the Task is Known: This case can be applied to a real-world job work where the task remains the same and only the variable changes. For example: examining a company's progress using the same visualization settings and same targets but with the new dataset or a newly hired data analyst trying to understand relationships between different variables using past reports. In other words, when the start and end points are known, and the path between these points is unknown.

Here, when the task is known, then displaying task-related help is possible and can be valuable. In this case, the system can direct the user to the right path and help the user to solve a particular task. Chances of recommendation success are higher as system knows the form of final expected result but the suggestions not generalizable as the system is designed to solve a particular type of task and can only generate recommendations which are related to that dataset type and task.

When the Task is Unknown: Since the task is unknown, the system also doesn't know how it can be solved or what to expect in the end. Therefore, the system can only show general suggestions. For example, a system could make a dataset suggestion to show outliers, or to reveal data similar to the data currently in the view. These data driven suggestions may be applicable to many tasks. In this case, the recommendations can only help users to understand the interface or dataset options and explore the possibilities, but the support may or may not direct them to the right path for solving the task. Here, the recommendations are more generalizable as they are task independent but

are not direct.

When Logs are Available: Here, the system uses data from previous users to decide on a recommendation for the current user in a particular scenario. For example, if most of the previous users felt confusion in the beginning and took a similar approach to overcome the confusion then for the next person, the system will analyze this trend and generate a recommendation based on the typical approach. This case is task independent and analysis the trends from the past logs to generate a recommendation. I believe that this is the best option because it increases the chances of generating a meaningful recommendation, and also, the recommendations can be generalizable.

In summary, I discussed and demonstrated some ideas about how the three key steps, "when, what and how", can be answered using emotions as feedback. Also, I explored the intervention design space and went over the possible cases to generate automatic recommendations.

Chapter 7

Conclusion

This chapter will summarize this thesis by discussing the contribution, the limitations of the work followed by some open-ended ideas for future projects. The goal of this work was to develop a recommendation system for visual analytic tasks which could leverage a mixed-initiative interaction approach for generating meaningful recommendations to sustain engagement. For that, I used bio-signals of negative emotions as implicit feedback to improve the human-computer interaction and make the overall process bi-directional. In other words, a user exhibits negative emotion (frustration, confusion, or anger) while working on a visual analytic task and the system detects those emotions and reacts accordingly. Note that, I only considered extreme negative emotion such as frustration, confusion and anger. Other negative emotions like fear, disgust and sadness were out of the scope of this thesis.

7.1 Contributions

As mentioned in the introduction chapter, there are three main contributions of this work. I will review each of them to show how I fulfilled those require-

ments.

Selecting least intrusive bio-sensors for recording user's emotional responses: For detecting emotions, I needed to use some bio-sensors so that I could record the users' body signals while they perform a visual analytic task and later use them for developing a negative emotion detection classifier. After a thorough investigation, I selected GSR and eye tracker for recording users' body signals. The GSR was responsible for calculating arousal, and the eye tracker was to measure the valence. Also, these two devices didn't require any difficult setup or calibration. The GSR was simply worn like a wristwatch and didn't require any calibration whereas the eye tracker was placed below the participant's monitor and only needed a 10s calibration. Out of all the available devices in my knowledge, I found the combination of these two devices was sufficient for detecting negative emotions, like frustration, and were least distracting.

Making an on-the-fly negative emotion detector: After selecting the sensors, I designed a user study for collecting the data from 28 participants. Again, the data was to train the machine learning classifier. Before building an online model which would detect emotion in real-time, I first tested the dataset offline where I used Matlab and Python for data processing, feature extraction and building the classifier. I tested the dataset with various supervised machine learning models and compared the recall score. Out of all the classifiers, the random forest model achieved the highest recall score, and was able to differentiate between negative emotion and normal state 88% of the time. Next, I optimized the model by selecting the seven most contributing features. After that, I merged all the operation functions in Python and used UDP socket programming to transfer the data in real-time. I simulated data from the participants to compare the accuracy of the online model with the offline model. The final online model was predicting an emotion in every 4s.

Solving the three key points to generate a recommendation — "when, what and how": After building a real-time negative emotion detection classifier, the

last step was to use this information to answer when to show a intervene, what help to show based on the context and how to show the recommendation. For that, I first created a communication bridge between the PivotSlice interface and the machine learning model using UDP socket programming. Next, I designed a rule-based approach which uses the predicted emotions to decide when to show help. Moreover, I used gaze location information for detecting the context, interface based or dataset based, for deciding what kind of help should be provided. Lastly, I explored various intervention styles to answer how a recommendation should be displayed while also considering the emotional intensity and the intrusion level.

7.2 Limitations

While the final proposed model demonstrated the idea of creating a mixed-initiative interaction using emotions and I included different verification steps, the project still builds on some assumptions and biases which I am going to discuss in this part. Firstly, the experiment was conducted in a controlled environment where the lighting condition was constant. This step was necessary because pupils are very sensitive towards change in lights and any variation in pupil size due to the lighting condition would have altered the dataset values. Next, the final dataset was consisting of 26 males and 2 females which results in unbalanced gender ratio. Emotional responses work differently for men and women [22], and therefore, this gender bias in the dataset could affect the results. Also, it was difficult to prove the high amount of false positive detection rate as I could only hypothesize based on some observations in the dataset but couldn't confirm.

Furthermore, it was difficult to tell if the participants were feeling a negative emotion because of the tasks or any other personal reasons. This limitation is unavoidable as for privacy reasons I did not ask participants about their

personal life before the experiment. The participants were informed that they could withdraw their names from the study anytime (during or after) for any reason.

I used the oversampling technique to generate artificial data for balancing the classes in the final dataset. Although the oversampling technique is widespread in the machine learning field, still every individual exhibits a different range of body signals and emotions. The artificial data cannot account for every case and therefore, cannot be as accurate as real data. Lastly, the design of appropriate interventions was not tested as a follow-up study was outside the scope of this thesis.

7.3 Future Work

Since there are some assumptions and limitations in the project, these gaps can be further studied and updated in the future work. The first task can be to compute and compare the results with a larger dataset which is gender balanced. The results could open some exciting paths such as comparing the emotional response between males and females. Moreover, comparing the results based on professional background, expertise, type of the task, or testing the new dataset with neural networks could also bring out some trends. Also, instead of using oversampling, it may be better to apply undersampling on the larger dataset to avoid artificial data in training.

Furthermore, the intervention chapter can be extended to a full-sized thesis project which could include: (1) using machine learning approach in place of weight distribution method (rule-based technique) for finding the intensity of an emotion. The machine learning models could make the system more flexible towards any unexpected cases. (2) Exploring the disengagement context into further parts and study how the dialogue-based recommendations can be helpful. (3) Identifying different types of negative emotions from the

sequence of predicted output such as confusion, anger or frustration. Right now, I am calculating different weights for each sequence, and these weights might be helpful in understanding various negative emotions. (4) Testing the final recommendations system and the intrusion level on the users.

7.4 Conclusion

This thesis argued that recommendation systems need to understand user's side to answer "when, what and how" to intervene as well as to sustain engagement. The existing guidance systems use implicit or explicit or hybrid signals as feedback to learn about the user but have failed to understand user's feelings. Here, I demonstrated an idea of classifying negative emotions, as detected through physiological signals, and used them as implicit feedback for a better understanding of the user's mental state to generate useful recommendations while also monitoring the intrusion level. Moreover, I explored this concept to fill the gaps in the existing recommendation techniques with affective computing. Finally, I successfully built a working recommendation model in visual analytic which uses detected emotions for deciding how to provide the guidance. This project contributes to the better understanding of mixed-initiative interactions and opens up ample of future research opportunities.

Bibliography

[1] AFERGAN, D., PECK, E. M., SOLOVEY, E. T., JENKINS, A., HINCKS, S. W., BROWN, E. T., CHANG, R., AND JACOB, R. J. Dynamic difficulty using brain metrics of workload. In *Proc. Conf. on Human Factors in Computing Systems* (2014), ACM, pp. 3797–3806.

[2] BIXLER, R., AND D'MELLO, S. Toward fully automated person-independent detection of mind wandering. In *International Conference on User Modeling, Adaptation, and Personalization* (2014), Springer, pp. 37–48.

[3] CALVO, R. A., AND D'MELLO, S. Affect detection: An interdisciplinary review of models, methods, and their applications. *IEEE Transactions on Affective Computing 1*, 1 (2010), 18–37.

[4] CERNEA, D., EBERT, A., AND KERREN, A. A study of emotion-triggered adaptation methods for interactive visualization. In *UMAP Workshops* (2013).

[5] CHAWLA, N. V., BOWYER, K. W., HALL, L. O., AND KEGELMEYER, W. P. SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research 16* (2002), 321–357.

[6] COLLINS, C., CARPENDALE, S., AND PENN, G. Docuburst: Visualizing document content using language structure. *Computer Graphics Forum*

(Proc. of the Eurographics/IEEE-VGTC Symposium on Visualization (EuroVis)) 28, 3 (2009), 1039–1046.

[7] CONATI, C., HOQUE, E., TOKER, D., AND STEICHEN, B. When to adapt: Detecting user's confusion during visualization processing. In *UMAP Workshops* (2013).

[8] DEL OLMO, F. H., AND GAUDIOSO, E. Evaluation of recommender systems: A new approach. *Expert Systems with Applications 35*, 3 (2008), 790–804.

[9] D'MELLO, S., OLNEY, A., WILLIAMS, C., AND HAYS, P. Gaze tutor: A gaze-reactive intelligent tutoring system. *International Journal of Human-Computer Studies 70*, 5 (2012), 377–398.

[10] DRISKELL, J. E., AND SALAS, E. *Stress and Human Performance*. Psychology Press, 2013.

[11] FRIDMAN, L., REIMER, B., MEHLER, B., AND FREEMAN, W. T. Cognitive load estimation in the wild. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2018), vol. 652, ACM, pp. 1–9.

[12] FRIENDLY, M. A brief history of data visualization. In *Handbook of Data Visualization*. Springer, 2008, pp. 15–56.

[13] GENA, C. Methods and techniques for the evaluation of user-adaptive systems. *The Knowledge Engineering Review 20*, 1 (2005), 1–37.

[14] GOTZ, D., AND WEN, Z. Behavior-driven visualization recommendation. In *Proceedings of the International Conference on Intelligent User Interfaces* (2009), ACM, pp. 315–324.

[15] HANSTEEN-IZORA, M., KULKARNI, A., AND STONE, A. Fostering trust in virtual teams. Tech. rep., Stanford University, 2002.

[16] HERNANDEZ-DEL OLMO, F., GAUDIOSO, E., AND BOTICARIO, J. G. Evaluating the intrusion cost of recommending in recommender systems. In *International Conference on User Modeling* (2005), Springer, pp. 342–346.

[17] ISENBERG, P., HEIMERL, F., KOCH, S., ISENBERG, T., XU, P., STOLPER, C. D., SEDLMAIR, M. M., CHEN, J., MÖLLER, T., AND STASKO, J. Vispubdata.org: A metadata collection about IEEE Visualization (VIS) publications. *IEEE Transactions on Visualization and Computer Graphics 23* (2017), 2199–2206.

[18] ISINKAYE, F., FOLAJIMI, Y., AND OJOKOH, B. Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal 16*, 3 (2015), 261–273.

[19] JAQUES, N., CONATI, C., HARLEY, J. M., AND AZEVEDO, R. Predicting affect from gaze data during interaction with an intelligent tutoring system. In *International Conference on Intelligent Tutoring Systems* (2014), Springer, pp. 29–38.

[20] KEIM, D., ANDRIENKO, G., FEKETE, J.-D., GÖRG, C., KOHLHAMMER, J., AND MELANÇON, G. Visual analytics: Definition, process, and challenges. In *Information Visualization*. Springer, 2008, pp. 154–175.

[21] KIM, J., AND ANDRÉ, E. Emotion recognition based on physiological changes in music listening. *IEEE Transactions on Pattern Analysis and Machine Intelligence 30*, 12 (2008), 2067–2083.

[22] KOCH, K., PAULY, K., KELLERMANN, T., SEIFERTH, N. Y., RESKE, M., BACKES, V., STÖCKER, T., SHAH, N. J., AMUNTS, K., KIRCHER, T., ET AL. Gender differences in the cognitive control of emotion: An fMRI study. *Neuropsychologia 45*, 12 (2007), 2744–2754.

[23] KRISCH, J. A. Why you hated clippy, that annoying microsoft paperclip, 2016. https://www.vocativ.com/319986/clippy-digital-assistant/index.html, last checked on 2018-06-08.

[24] KURNIAWAN, H., MASLOV, A. V., AND PECHENIZKIY, M. Stress detection from speech and galvanic skin response signals. In *26th International Symposium on Computer-Based Medical Systems (CBMS)* (2013), IEEE, pp. 209–214.

[25] LALLÉ, S., TOKER, D., CONATI, C., AND CARENINI, G. Prediction of users' learning curves for adaptation while using an information visualization. In *The 20th International Conference on Intelligent User Interfaces* (2015), ACM, pp. 357–368.

[26] LIU, Y., SOURINA, O., AND NGUYEN, M. K. Real-time EEG-based human emotion recognition and visualization. In *International Conference on Cyberworlds (CW)* (2010), IEEE, pp. 262–269.

[27] LU, Y., ZHENG, W.-L., LI, B., AND LU, B.-L. Combining eye movements and EEG to enhance emotion recognition. In *IJCAI* (2015), vol. 15, pp. 1170–1176.

[28] MCCUAIG, J., PEARLSTEIN, M., AND JUDD, A. Detecting learner frustration: towards mainstream use cases. In *International Conference on Intelligent Tutoring Systems* (2010), Springer, pp. 21–30.

[29] OLSSON, P. Real-time and offline filters for eye tracking. Master's thesis, KTH Electrical Engineering, 2007.

[30] PARTALA, T., AND SURAKKA, V. Pupil size variation as an indication of affective processing. *International Journal of Human-Computer Studies 59*, 1-2 (2003), 185–198.

[31] PEIRCE, N., CONLAN, O., AND WADE, V. Adaptive educational games: Providing non-invasive personalised learning experiences. In *International*

Conference on Digital Games and Intelligent Toys Based Education (2008), IEEE, pp. 28–35.

[32] PEKRUN, R., GOETZ, T., TITZ, W., AND PERRY, R. P. Academic emotions in students' self-regulated learning and achievement: A program of qualitative and quantitative research. *Educational Psychologist 37*, 2 (2002), 91–105.

[33] PICARD, R. W., ET AL. *Affective Computing*. Perceptual Computing Section, Media Laboratory, Massachusetts Institute of Technology, 1995.

[34] SHARMA, N., AND GEDEON, T. Objective measures, sensors and computational techniques for stress recognition and classification: A survey. *Computer Methods and Programs in Biomedicine 108*, 3 (2012), 1287–1301.

[35] SHIMMER GSR. Measuring human emotion: Reactions to media stimuli, 2015. `https://www.shimmersensing.com/assets/images/content/case-study-files/Emotional/_Response/_27July2015.pdf`, last checked on 2018-05-21.

[36] SIVAPALAN, S., SADEGHIAN, A., RAHNAMA, H., AND MADNI, A. M. Recommender systems in e-commerce. In *World Automation Congress (WAC)* (2014), IEEE, pp. 179–184.

[37] STAFF. Comparison shopping tools: Find the best offers fresh approach, 2012. `http://www.sub5zero.com/comparison-shopping-tools-findthebest-offers-fresh-approach/`, last checked on 2018-06-020.

[38] STEICHEN, B., CONATI, C., AND CARENINI, G. Inferring visualization task properties, user performance, and user cognitive abilities from eye gaze data. *Transactions on Interactive Intelligent Systems (TiiS) 4*, 2 (2014), 1–29.

[39] STEICHEN, B., WU, M. M., TOKER, D., CONATI, C., AND CARENINI, G. Te, te, hi, hi: Eye gaze sequence analysis for informing user-adaptive information visualizations. In *International Conference on User Modeling, Adaptation, and Personalization* (2014), Springer, pp. 183–194.

[40] SUN, F.-T., KUO, C., CHENG, H.-T., BUTHPITIYA, S., COLLINS, P., AND GRISS, M. Activity-aware mental stress detection using physiological sensors. In *International Conference on Mobile Computing, Applications, and Services* (2010), Springer, pp. 282–301.

[41] TURBAN, E., OUTLAND, J., KING, D., LEE, J. K., LIANG, T.-P., AND TURBAN, D. C. Intelligent (smart) e-commerce. In *Electronic Commerce*. Springer, 2018, pp. 249–283.

[42] VARTAK, M., HUANG, S., SIDDIQUI, T., MADDEN, S., AND PARAMESWARAN, A. Towards visualization recommendation systems. *Special Interest Group on Management of Data Record 45*, 4 (2017), 34–39.

[43] VOIGT, M., PIETSCHMANN, S., GRAMMEL, L., AND MEISSNER, K. Context-aware recommendation of visualization components. In *The Fourth International Conference on Information, Process, and Knowledge Management (eKNOW)* (2012), pp. 101–109.

[44] WANG, L., WANG, G., AND ALEXANDER, C. A. Big data and visualization: methods, challenges and technology progress. *Digital Technologies 1*, 1 (2015), 33–38.

[45] WHIGHAM, N. Microsoft's clippy was one of the most hated features ever, but distance makes the heart grow fonder, 2017. `https://www.news.com.au/technology/innovation/design/microsofts-clippy-was-one-of-the-most-hated-features-ever-but-distance-makes-the-heart-grow-fonder/news-story/772605472436e0f6860fd04e4cf1b6dd`, last checked on 2018-06-08.

[46] WIOLETA, S. Using physiological signals for emotion recognition. In *The 6th International Conference on Human System Interaction (HSI)* (2013), IEEE, pp. 556–561.

[47] WONGSUPHASAWAT, K., MORITZ, D., ANAND, A., MACKINLAY, J., HOWE, B., AND HEER, J. Voyager: Exploratory analysis via faceted browsing of visualization recommendations. *IEEE Transactions on Visualization and Computer Graphics*, 1 (2016), 649–658.

[48] WOOLF, B. P. *Building Intelligent Interactive Tutors: Student-Centered Strategies for Revolutionizing E-Learning*. Morgan Kaufmann, 2010.

[49] ZHAI, J., BARRETO, A. B., CHIN, C., AND LI, C. Realization of stress detection using psychophysiological signals for improvement of human-computer interactions. In *SoutheastCon* (2005), IEEE, pp. 415–420.

[50] ZHAO, J., COLLINS, C., CHEVALIER, F., AND BALAKRISHNAN, R. Interactive exploration of implicit and explicit relations in faceted datasets. *IEEE Transactions on Visualization and Computer Graphics 19*, 12 (2013), 2080–2089.

[51] ZHENG, W.-L., DONG, B.-N., AND LU, B.-L. Multimodal emotion recognition using EEG and eye tracking data. In *36th Annual International Conference of the Medicine and Biology Society (EMBC)* (2014), IEEE, pp. 5040–5043.